Supercomputing Frontiers and Innovations

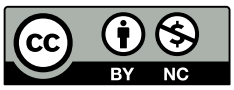
2017, Vol. 4, No. 4

Scope

- Enabling technologies for high performance computing
- Future generation supercomputer architectures
- Extreme-scale concepts beyond conventional practices including exascale
- Parallel programming models, interfaces, languages, libraries, and tools
- Supercomputer applications and algorithms
- Distributed operating systems, kernels, supervisors, and virtualization for highly scalable computing
- Scalable runtime systems software
- Methods and means of supercomputer system management, administration, and monitoring
- Mass storage systems, protocols, and allocation
- Energy and power minimization for very large deployed computers
- Resilience, reliability, and fault tolerance for future generation highly parallel computing systems
- Parallel performance and correctness debugging
- Scientific visualization for massive data and computing both external and in situ
- Education in high performance computing and computational science

Editorial Board

Contents



DOI: 10.14529/jsfi170401

Towards A Data Centric System Architecture: SHARP

Gil Bloch[1], *Devendar Bureddy*[2], *Richard L. Graham*[3], *Gilad Shainer*[2], *Brian Smith*[3]

Increased system size and a greater reliance on utilizing system parallelism to achieve computational needs, requires innovative system architectures to meet the simulation challenges. The SHARP technology is a step towards a data-centric architecture, where data is manipulated throughout the system. This paper introduces a new SHARP optimization, and studies aspects that impact application performance in a data-centric environment. The use of UD-Multicast to distribute aggregation results is introduced, reducing the letency of an eight-byte *MPI_Allreduce()* across 128 nodes by 16%. Use of reduction trees that avoid the inter-socket bus further improves the eight-byte *MPI_Allreduce()* latency across 128 nodes, with 28 processes per node, by 18%. The distribution of latency across processes in the communicator is studied, as is the capacity of the system to process concurrent aggregation operations.

Keywords: data centric architecture, SHARP, collectives, MPI.

Introduction

The challenge of providing increasingly unprecedented levels of effective computing cycles, for tightly coupled computer-based simulations, continues to pose new technical hurdles. With each hurdle traversed, a new challenge comes to the forefront, with many architectural features emerging to address these problems. This has included the introduction of vector compute capabilities to single processor systems, such as the CDC Star-100 [28] and the Cray-1 [27], followed by the introduction of small-scale parallel vector computing, such as the Cray-XMP [5], custom-processor-based tightly-coupled MPPs, such as the CM-5 [21] and the Cray T3D [17], followed by systems of clustered commercial-off-the-shelf micro-processors, such as the Dell PowerEdge C8220 Stampede at TACC [30] and the Cray XK7 Titan computer at ORNL [24]. For a decade or so the latter systems relied mostly on Central Processing Unit (CPU) frequency upticks to provide the increase in computational power. But, as a consequence of the end of Dennard scaling [9], the single CPU frequency has plateaued, with contemporary HPC cluster performance increases depending on rising numbers of compute engines per silicon device to provide the desired computational capabilities. Today HPC systems use many-core host elements that utilize, for example, X86, Power, or ARM processors, General Purpose Graphical Processing Units (GPGPUs) and Field Programmable Gate Arrays (FPGAs), [15], to keep scaling the system performance. Network capabilities have also increased dramatically over the same period, with changes such as increases in bandwidth, decreases in latency, and communication technologies like InfiniBand RDMA that offload processing from the CPU to the network.

With increasing compute engine counts, system architectures have continued to be CPU centric, with these system elements being involved in the vast majority of data manipulation. This has resulted in unnecessary data movement and undesirable competition between computational, communication, storage and other needs for the same computational resources. A Data-Centric system architecture, which co-locates computational resources and data throughout the system, enables data to be processed all across the system, and not only by CPU's

[1]Mellanox Technologies, Inc., Yokneam, Israel
[2]Mellanox Technologies, Inc., Sunnyvale, California, U.S.
[3]Mellanox Technologies, Inc., Knoxville, Tennessee, U.S.

at the edge. For example, data can be manipulated as it is being transferred within the data center network as part of a collective operation. This type of approach addresses latency and other performance bottlenecks that exist in the traditional CPU-Centric architecture. Mellanox focuses on CPU offload technologies designed to process data as it moves through the network, either by the Host Channel Adapter (HCA) or the switch. This frees up CPU cycles for computation, reduces the amount of data transferred over the network, allows for efficient pipelining of network and computation, and provides for very low communication latencies. To accomplish a marked increase in application performance, there has been an effort to optimize often used communication patterns, such as collective operations, in addition to the continuous improvements to basic communication metrics, such as point-to-point bandwidth, latency, and message rate.

InfiniBand technologies are being transformed to support such data-centric system architectures. These include technologies such as SHARP for handling data reduction and aggregation, hardware-based tag matching and Network data hardware-gather scatter capabilities. These technologies are used to process data and network errors at the network levels, without the need for data to reach a CPU, reducing overall volume of transferred data and system resilience.

This paper extends the investigation of the the SHARP technology previously introduced [12] for offloading aggregation and reduction operations to InfiniBand switches. The paper is organized as follows: Section 1 presents previous related work in offload technologies. Section 2 describes the new UD-Multicast protocol which utilizes multiple children in the reduction tree to avoid using internal node interconnect between sockets. Section 3 describes the benchmarks and applications investigated, and discusses the distribution of latencies across processes in a communicator and the network's ability to process multiple reduction operations concurrently. The final section provides a summary and discussion of the work presented.

1. Previous Work

In the past extensive work has been done on improving performance of blocking and non-blocking barrier and reduction algorithms.

Algorithmic work performed by Venkata et al. [33] developed short vector blocking and non blocking reduction and barrier operations using a recursive K-ing type host-based approach, and extended work by Thakur [31]. Vadhiar et al. [32] presented implementations of blocking reduction, gather and broadcast operations using sequential, chain, binary, binomial tree and Rabenseifner algorithms. Hoefler et al. [16] studied several implementations of nonblocking *MPI_Allreduce()* operations, showing performance gains when using large communicators and large messages.

Some work aimed to optimize collective operations for specific topologies. Representative examples are ref. [6] and [22], which optimized collectives for mesh topologies, and for hypercubes, respectively.

Other work presented hardware support for performance improvement. Conventionally, most implementations use the CPU to setup and manage collective operations, with the network just used as a data conduit. However, Quadrics [26] implemented support for broadcast and barrier in network device hardware. Recently IBM's Blue Gene supercomputer included network-level hardware support for barrier and reduction operations. Its preliminary version Blue Gene/L [11] which uses torus interconnect [1], provided up to twice throughput performance gain of all-to-all collective operations [2, 20]. On a 512 node system the latency of the 16 byte *MPI_Allreduce()* the

latency was 4.22 μ-seconds. Later, a message passing framework DCMF for the next-generation supercomputer Blue Gene/P was introduced [18]. MPI collectives optimization algorithms for this generation of Blue Gene were analyzed in [10]. The recent version Blue Gene/Q [14] provides additional performance improvements for MPI collectives [19]. On a 96,304 node system, the latency of a short allreduce is about 6.5 μ-seconds. IBM's PERCS system [4] fully offloads collective reduction operations to hardware. Finally, Mai et al. presented the NetAgg platform [23], which uses in-network middleboxes for partition/aggregation operations, to provide efficient network link utilization. Cray's Aries network [3] implemented 64 byte reduction support in the HCA, supporting reduction trees with a radix of up to 32. The eight byte *MPI_Allreduce()* latency for about 12,000 process with 16 processes per host was close to ten u-seconds.

Several APIs have been proposed for offloading collective operation management to the HCA. This includes the Mellanox's CORE-Direct [13], protocol, Portal 4.0 triggered operations [7], and an extension to Portals 4.0 [29]. All these support protocols that use end-point management of the collective operations, whereas in the current approach the end-points are involved only in collective initiation and completion, with the switching infrastructure supporting the collective operation management.

2. Aggregation Protocol

A goal of the new network co-processor architecture is to optimize completion time of frequently used global communication patterns and to minimize their CPU utilization. The first set of patterns being targeted are global reductions of short vectors, and include barrier synchronization, and small data reductions. As previously mentioned, the SHARP protocol has already been described in detail, therefore, only a brief description is provided in this section, highlighting the new hardware capability that is introduced.

SHARP provides an abstraction describing data reduction and aggregation. The protocol defines aggregation nodes (ANs) which form the nodes of a reduction tree. These trees overlay a physical network. Figure 1 shows an example of a physical network topology, with Fig. 2 describing a possible reduction tree constructed over this physical topology. The aggregation nodes are colored in red, with the leaves of the tree, the blue stars, being source of the data.

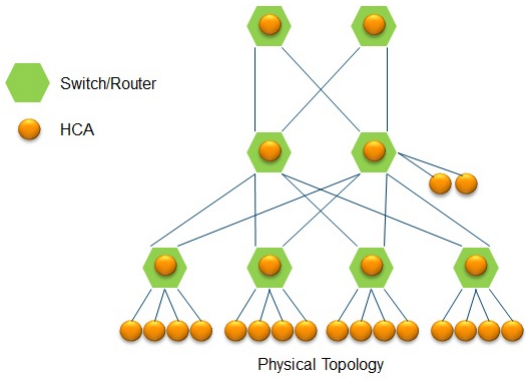


Figure 1. Physical Network Topology

Aggregation operations are defined for SHARP groups. These groups are formed as subtrees of SHARP trees, where multiple groups may be formed from a given SHARP tree. Figure 3 gives an example of a SHARP group of size eight.

An aggregation operation is performed with participation of each member of the aggregation group. To initiate such an operation, members of the aggregation group send their aggregation request message to their leaf aggregation node. The aggregation request header contains all needed information to perform the aggregation, and includes the data description, i.e. the data type, data size, and number of such elements, and the aggregation operations to be performed, such as a min or sum operation. An aggregation node receiving aggregation requests collects these from all its children and performs the aggregation operation once all the expected requests arrive. The root aggregation node performs the final aggregation producing the result of the aggregation operation.

This aggregation result is distributed in up to two of several possible ways. The destination may be one of several targets, including one of the requesting processes, such as in the case of *MPI_Reduce()*, all the group processes, such as in the case of an *MPI_Allreduce()* operation, or a separate process that may not be a member of the reduction group. An aggregation tree can be used to distribute the data in these cases.

The new hardware capability described in this paper is that the target may also be a user-defined InfiniBand multicast address. It is important to note that while multicast data distribution is supported by the underlying transport, it provides an unreliable delivery mechanism. Any reliability protocol needed must be provided on top of this mechanism.

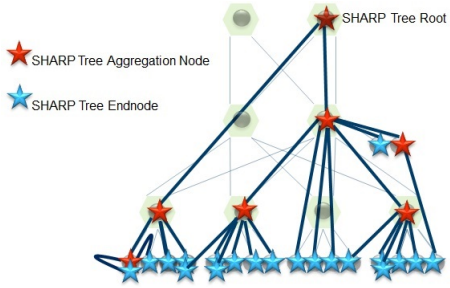


Figure 2. Logical SHARP Tree. Note that in the SHARP abstraction an Aggregation Node may be hosted by an end-node

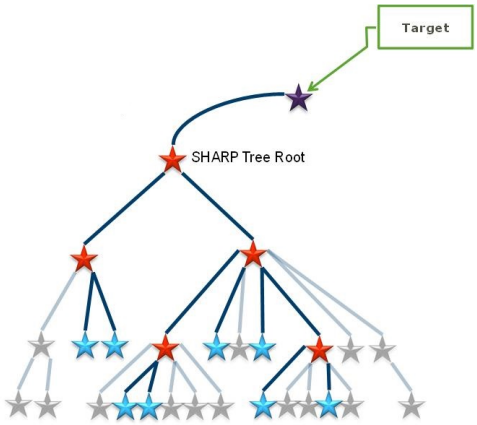


Figure 3. A SHARP group defined for the SHARP tree. The red stars designate AN's and the blue stars the tree leaves

The protocol does not define the data transport, so that communication between AN's can occur using a range of transports, such as RDMA-enabled protocols like InfiniBand or RDMA over Converged Ethernet (RoCE). It also does not handle packet loss or reordering, requiring a reliable transport which provides reliable in-order delivery of packets to the upper layer.

2.1. SwitchIB-2-Based Aggregation Support

In the SwitchIB-2 implementation, the aggregation node logic is implemented as an InfiniBand TCA integrated into the switch ASIC. The transport used for communication between ANs and between AN and hosts in the aggregation tree is the InfiniBand Reliable Connection (RC) transport. The results are distributed from the root to the leaf nodes, or hosts, down the tree, or to a target InfiniBand Multicast group.

The aggregation node implementation includes a high performance Arithmetic Logic Unit (ALU), used to perform the aggregation operations supported by the aggregation node. It can operate on 32- and 64-bit signed and unsigned integers and floating point data. The supported operations include sum, min and max, MPIs MinLoc and MaxLoc, bitwise OR, AND, and XOR, which include all the operations, with the exception of the product, needed to support the MPI standard and the OpenSHMEM specification.

Requests are collected in the TCA, with the reduction performed only after all operands are available, in a predetermined and fixed order. SwitchIB-2 implements a predictable operation ordering to enable repeatable results regardless of the order of arrival of the aggregation requests.

When using hardware multicast to distribute the aggregation results, the result also needs to be distributed with a reliable protocol to ensure delivery of these results.

3. Benchmark Results

To evaluate the SHARP capabilities, both low-level MPI benchmarks, as well as an application level benchmark are used.

A 128 host system is used for these experiments. Each node has two 14-core Broadwell CPUs running at 2.60 GHz, with 256GB of RAM memory. ConnectX-4 HCAs are used running at 100Gb/s. The fabric uses a two-level fat-tree with SwitchIB-2 switches and eight leaf switches, each connecting to 16 hosts. The hosts run RedHat Linux 7.2, and the tests were carried out with OFED 3.4-2.1.9.0. A pre-release version of HPC-X, the Mellanox supported MPI, is used, which includes a set of MPI collective routines that access and use the SHARP hardware capabilities, embedded in the SwitchIB-2 switches, to optimize the performance of the corresponding MPI collectives.

3.1. MPI-Level SHARP Measurements

The OSU *MPI_Allreduce()* test [25] is used to measure the SHARP latency.

Figure 4 shows the latency of *MPI_Allreduce()* operations as a function of message size and the mode of result distribution, with one process per-node. Using UD multicast for distributing the result takes advantage of the O(1) multicast capabilities for improved performance, but is unreliable (bit error rate being on the order of 10^{-15}) requiring the additional RC result distribution to provide the result when a UD packet is dropped. Using UD multicast and RC to distribute the results improves latency in the range of 15-58% relative to using RC only for

this distribution, even with the duplicate result distribution. The improvement relative to the host-based approach is in the range of 143 to 385 percent.

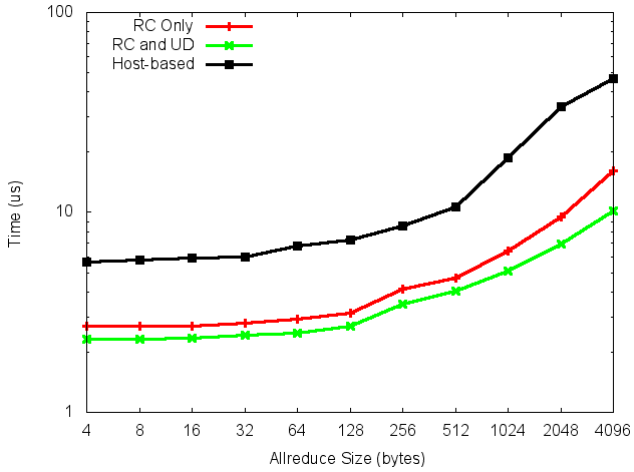


Figure 4. 128 node *MPI_Allreduce()* average latency with different modes of result distribution. A comparison to the host-only algorithm is also included. Latency is reported in μ-seconds

SHARP reduction trees assume some sort of host-level aggregation prior to sending data to the leaf AN, because of the limitation on AN's radix. Figure 5 shows the latency of the *MPI_Allreduce()* operation when using one connection per socket (2 channels) into the SHARP reduction tree, avoiding reduction over the internal chip network, and one connection per node (1 channel). As the results show, for messages up to 1024 bytes in size, this reduces latency by more than ten percent. With larger messages, an increase in latency is observed. The two-channel case eliminates the host-side intra-socket reduction steps, it increases the leaf AN radix by a factor of two. As the vector length increases, this manifests itself with a larger latency relative to the one-channel case.

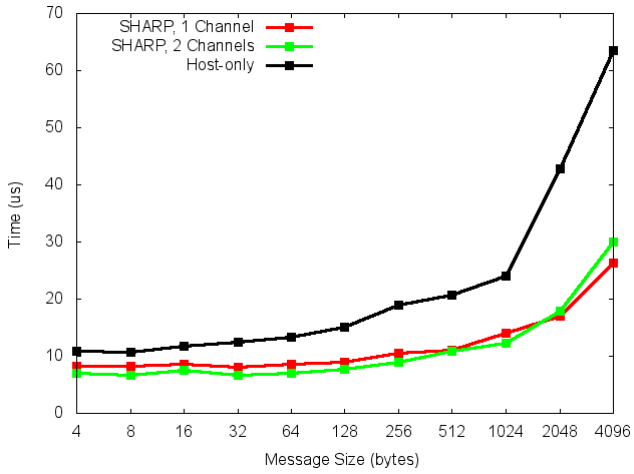


Figure 5. 128 node, 28 processes per-node *MPI_Allreduce()* average latency in μ-seconds

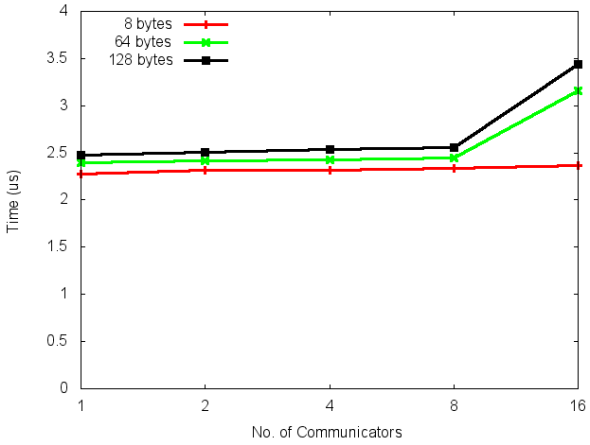
To get a better understanding of the spread in completion times across the communicator, several metrics are collected to characterize this behavior. Table 1 lists the average *MPI_Allreduce()* latencies, along with quartile data, minimum value and maximum value to describe the data distribution, using UD-multicast for result distribution, and one process per node. These are reported for the average of the full collective operation (measured as as the average of the

collective operation) and for the the completion of each of the individual ranks in the communicator. As expected, there is greater variance in individual completion times, as compared with the average per-collective completion time. Also, we see that the SHARP based collectives have a much smaller per-rank latency range.

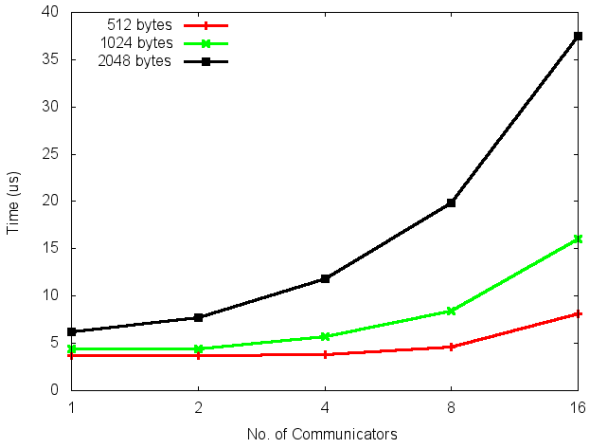
Table 1. *MPI_Allreduce()* Latency (μsec) Distribution of a 127 Node Cluster with One Process Per Node

	Size (B)	Ave.	Quartiles min, Q1, Q2, Q3, max Per Operation Average	Quartiles min, Q1, Q2, Q3, max Per Process Data
SHARP	4	2.39	2.38, 2.39, 2.39, 2.40, 2.41	2.16, 2.35, 2.37, 2.41, 13.49
Host		5.09	4.99, 5.07, 5.09, 5.10, 5.13	2.34, 4.74, 4.88, 5.13, 12.69
SHARP	8	2.39	2.37, 2.38, 2.39, 2.39, 2.40	2.14, 2.35, 2.37, 2.41, 2.78
Host		5.18	5.11, 5.18, 5.18, 5.19, 5.21	2.51, 4.83, 4.98, 5.20, 15.49
SHARP	16	2.41	2.40, 2.40, 2.41, 2.41, 2.42	2.20, 2.37, 2.39, 2.43, 2.90
Host		5.26	5.20, 5.25, 5.26, 5.28, 5.31	2.48, 4.91, 5.05, 5.27, 16.15
SHARP	32	2.47	2.47, 2.48, 2.48, 2.48, 2.49	2.26, 2.45, 2.47, 2.50, 3.02
Host		5.32	5.26, 5.31, 5.32, 5.33, 5.38	2.48, 4.97, 5.11, 5.36, 13.64
SHARP	64	2.55	2.55, 2.55, 2.56, 2.56, 2.57	2.26, 2.52, 2.55, 2.57, 3.00
Host		5.98	5.72, 5.98, 6.00, 6.00, 6.04	2.70, 5.65, 5.80, 6.01, 18.38
SHARP	128	2.76	2.75, 2.76, 2.76, 2.76, 2.77	2.44, 2.72, 2.74, 2.77, 10.30
Host		6.43	6.17, 6.43, 6.44, 6.45, 6.51	3.23, 6.10, 6.27, 6.50, 13.66
SHARP	256	3.52	3.51, 3.51, 3.52, 3.52, 3.53	3.04, 3.48, 3.51, 3.54, 7.37
Host		7.55	7.38, 7.54, 7.57, 7.58, 7.62	4.19, 7.29, 7.42, 7.63, 16.36
SHARP	512	4.10	4.07, 4.07, 4.07, 4.10, 4.25	3.63, 4.05, 4.08, 4.14, 10.70
Host		9.16	8.96, 9.14, 9.17, 9.19, 9.22	4.05, 8.93, 9.04, 9.21, 24.66
SHARP	1024	5.19	5.11, 5.15, 5.18, 5.21, 5.32	4.68, 5.07, 5.15, 5.28, 7.70
Host		18.49	16.24, 17.36, 18.27, 19.60, 20.52	11.38, 17.33, 18.67, 19.52, 31.69
SHARP	2048	7.55	7.52, 7.54, 7.55, 7.56, 7.58	5.61, 7.22, 7.51, 7.80, 16.65
Host		33.47	31.33, 32.56, 33.60,3 4.27, 36.89	28.83, 32.48, 33.49, 34.25, 50.40
SHARP	4096	12.34	12.30, 12.33, 12.34, 12.35, 12.39	10.38, 11.54, 11.99, 13.06, 17.27
Host		45.99	42.60, 45.10, 46.06, 46.80, 49.49	39.15, 44.99, 45.95, 46.89, 58.68

For the SHARP capabilities to be useful in a general purpose production system, where multiple jobs run concurrently, potentially sharing ANs, it is useful to study the systems ability to support concurrent SHARP operations. The system's capacity to service concurrent collective operations is studied by running multiple collective operations at the same time, using completely overlapping SHARP-tree groups. The OSU-latency test was modified to run concurrent collective operations with non-overlapping MPI Communicators, with the MPI process layout configured to achieve this overlap. As the results show in Fig. 6 for communicators of size eight, SHARP is able to accommodate many outstanding operations very well. Latency starts to degrade at a message size of 2048 bytes, with eight concurrent operations, where as many as sixty four operations are in flight. With sixteen concurrent operations, latency is impacted by about 30% with a message size of 64 bytes.



(a) Small-size Allreduce



(b) Large-size Allreduce

Figure 6. SHARP *MPI_Allreduce()* latency (in μ-seconds) for 128 nodes with varying simultaneous communicators

Table 2 presents the *MPI_Allreduce()* latency as a function of the number of outstanding SHARP operations each group is configured to allow. The eight byte data requires only one SHARP-level operation per MPI operation, whereas the 2048 byte reduction requires eight such operations. As expected, we see that the eight byte reduction is minimally impacted by the number of allowed outstanding SHARP operations, except in the eight communicator test, where there are insufficient resources for all communicators, and the test does not run as written. The 2048 byte MPI-level operation is negatively impacted by the lack of sufficient resources to pipeline the entire operation at once, but even with only two outstanding SHARP operations supported, there is the benefit of some pipelining, with the latency being less than four times that of the eight operation case.

3.2. Application Benchmarks

Table 3 shows the result of running the Algebraic Multi-Grid (AMG) [8] micro benchmark on 64 nodes, with 28 processes per node. The AMG benchmark uses an eight byte data reduction. On average, running five of the AMG test cases (Laplace, 27 point, Jumps, def/pool1 and def/pool0) an average improvement of 1.8% in total test run time was measured when using

Table 2. Eight Process *MPI_Allreduce()* Average Latency (μsec) as a Function of the Number of Communicators Operating in Parallel and as a Function of Maximum Outstanding SHARP Operations (OSOs) Available

8 bytes Total OSOs	OSOs per Comm	1 Comm	2 Comm	4 Comm	8 Comm
8	1	2.24	2.263	2.260	N/A
16	1	2.210	2.253	2.260	2.238
32	2	2.210	2.245	2.250	2.236
2048 bytes OSOs	**OSOs per Comm**	**1 Comm**	**2 Comm**	**4 Comm**	**8 Comm**
32	2	11.890	11.990	12.154	14.991
64	4	7.695	7.783	8.374	12.229
128	8	5.495	5.533	6.710	10.351

SHARP. The figure of merit used is system-size * number-of-iterations / (solve-time in units of seconds).

Table 3. AMG Figure-of-Merit (Higher is Better) Data for Five Different Tests, Run on 64 Nodes with 28 Processes Per Node, and a System Configured with Low System Noise

Job Type	Laplace	27pt	Jumps	Pooldist_1	Pooldist_0
non-SHARP	2.10E+09	1.64E+09	2.84E+09	2.45E+09	3.82E+09
SHARP	2.21E+09	1.68E+09	2.86E+09	2.40E+09	3.92E+09
% Change	5.2	2.4	0.7	-2.0	2.6

Discussion and Conclusions

To improve MPI-level aggregation performance, UD-Multicast is used to distribute results from the root of the aggregation-tree, and SHARP trees that avoid using the host's inter-socket bus for aggregations are employed. Employing UD-multicast to distribute the aggregated values reduces overall operation latency, even though the result is sent twice to ensure reliable delivery, once using UD-Multicast and once with RC. The UD packets allow for fast result distribution, with a very low packet-rate loss. The RC packets sent to ensure data delivery arrive a little later, and impact latency only when the UD packets are lost. This improves eight byte reduction at 128 nodes by 16%, and the 4096 byte latency by 58%. The distribution using UD-Multicast benefits from the switch's ability to replicate the data packet to all ports relevant to the multicast group in parallel, whereas the RC packet replication has some degree of serialization. In addition, for small message distribution message rate, rather than bandwidth, is the primary performance limiter. The high message rate, of 195 messages per port, per μ-second supported by the SwitchIB-2 device, is capable of handling the duplicated data.

Using the intra-host bus for data exchange between sockets can be expensive relative to the intra-socket communication. It is frequently more efficient to avoid using this bus when accessing the network, and therefore a similar approach has been investigated for SHARP reductions. As the results show, aggregating data on a per-socket basis also helps reduce operation latency for small message sizes, reducing the eight byte operation latency by 16% at 128 nodes. However, as the data size increases, competition for the PCIe bus bandwidth from the host to the network and the two fold increase in AN radix at the leaf switches make this particular optimization undesirable.

In general purpose data-centric environments, with multiple jobs running on the system at the same time jobs compete for a fixed set of resources, unless special care has been taken to isolate the resources used by separate jobs. In the case of SHARP the AN resources are an additional set of resources, beyond the host and other network resources that may be shared. The impact of such sharing on the SHARP latencies has been studied by running concurrent reductions on the same reduction-tree and limiting the number of concurrent aggregations.

To study the effectiveness of the protocol in a multi-job scenario, where some ANs may be used by multiple jobs, we ran up to sixteen concurrent collectives simultaneously. This is expected to be a worse-case type of scenario, because the test forces the collective operation concurrency. Since application runs typically are not synchronized, and they do more than just run collective operations, the impact on concurrent running applications using the same AN resources is expected to be less. The results show that the impact on the small message reduction latency is small, but as the message size increases the impact of this sharing becomes noticeable due to the competition for bandwidth. At 2048 byte message size and 128 nodes, a small impact is noticed when two operations are running concurrently, but with four it is still advantageous to use the SHARP protocol over the host-based protocol.

We also observed that when there are insufficient resources to pipeline a reduction operation with independent resources, there are still benefits to such optimization when compared with the host-based approach. A 2048 byte message size and 128 nodes requires eight OSOs for the full message reduction to be concurrently in flight. However, providing only two such OSOs still reduces the operation latency relative to the host-based approach.

Finally, collective operations are known to amplify application load imbalance. Looking at the per-process spread in collective operations, we see that the SHARP based collectives are less susceptible to imbalance within the collective algorithms themselves, thus supporting application scalability better than the host-based algorithms.

In conclusion, this paper has introduced the ability to use UD-multicast for aggregation result distribution and presented several aspects of the SHARP protocol not previously examined. Benchmark and application results show that the protocol is effective, and help to show how to best utilize the underlying SHARP capabilities in a general purpose data-centric environment.

References

1. Adiga, N.R., Blumrich, M.A., Chen, D., Coteus, P., et al.: Blue Gene/L torus interconnection network. IBM Journal of Research and Development 49(2/3), 265 (2005), DOI: 10.1147/rd.492.0265

2. Almási, G., Heidelberger, P., Archer, C.J., Martorell, X., Erway, C.C., Moreira, J.E., Steinmacher-Burow, B., Zheng, Y.: Optimization of MPI collective communication on BlueGene/L systems. In: Proceedings of the 19th annual international conference on Supercomputing. pp. 253–262. ACM (2005), DOI: 10.1145/1088149.1088183

3. Alverson, B., Froese, E., Kaplan, L., Roweth, D.: Cray XC series network. Tech. rep., Cray Inc. (2012), https://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf, accessed: 2017-10-01

4. Arimilli, B., Arimilli, R., Chung, V., Clark, S., Denzel, W., Drerup, B., Hoefler, T., Joyner, J., Lewis, J., Li, J., et al.: The PERCS high-performance interconnect. In: High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on. pp. 75–82. IEEE (2010), DOI: 10.1109/HOTI.2010.16

5. August, M.C., Brost, G.M., Hsiung, C.C., Schiffleger, A.J.: Cray X-MP: The birth of a supercomputer. Computer 22(1), 45–52 (1989), DOI: 10.1109/2.19822

6. Barnett, M., Littlefield, R.J., Payne, D.G., van de Geijn, R.A.: Global combine on mesh architectures with wormhole routing. In: The Seventh International Parallel Processing Symposium, Proceedings, Newport Beach, California, USA, April 13-16, 1993. pp. 156–162 (1993), DOI: 10.1109/IPPS.1993.262873

7. Barrett, B., Brightwell, R., Hemmert, S., Pedretti, K., Wheeler, K., Underwood, K.D., Reisen, R., Maccabe, A.B., Hudson, T.: The Portals 4.0 network programming interface, technical report SAND201210087. https://www.osti.gov/scitech/biblio/1088065 (2012), accessed: 2017-10-01

8. CORAL Collaboration: Benchmark codes. https://asc.llnl.gov/CORAL-benchmarks/#amg2013, accessed: 2017-10-01

9. Esmaeilzadeh, H., Blem, E., Amant, R.S., Sankaralingam, K., Burger, D.: Dark silicon and the end of multicore scaling. In: Computer Architecture (ISCA), 2011 38th Annual International Symposium on. pp. 365–376. IEEE (2011), DOI: 10.1145/2000064.2000108

10. Faraj, A., Kumar, S., Smith, B., Mamidala, A., Gunnels, J.: MPI collective communications on the Blue Gene/P Supercomputer: Algorithms and optimizations. In: High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on. pp. 63–72. IEEE (2009), DOI: 10.1109/HOTI.2009.12

11. Gara, A., Blumrich, M.A., Chen, D., Chiu, G.T., Coteus, P., Giampapa, M.E., Haring, R.A., Heidelberger, P., Hoenicke, D., Kopcsay, G.V., et al.: Overview of the Blue Gene/L system architecture. IBM Journal of Research and Development 49(2), 195–212 (2005), DOI: 10.1147/rd.492.0195

12. Graham, R.L., Bureddy, D., Lui, P., Rosenstock, H., Shainer, G., Bloch, G., Goldenerg, D., Dubman, M., Kotchubievsky, S., Koushnir, V., Levi, L., Margolin, A., Ronen, T., Shpiner, A., Wertheim, O., Zahavi, E.: Scalable hierarchical aggregation protocol (SHArP): A hardware architecture for efficient data reduction. In: Proceedings of the First Workshop on Optimization of Communication in HPC. pp. 1–10. COM-HPC '16, IEEE Press, Piscataway, NJ, USA (2016), DOI: 10.1109/COM-HPC.2016.6

13. Graham, R.L., Poole, S., Shamis, P., Bloch, G., Bloch, N., Chapman, H., Kagan, M., Shahar, A., Rabinovitz, I., Shainer, G.: Connectx-2 infiniband management queues: First investigation of the new support for network offloaded collective operations. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 53–62. CCGRID '10, IEEE Computer Society, Washington, DC, USA (2010), DOI: 10.1109/CCGRID.2010.9

14. Haring, R.A., Ohmacht, M., Fox, T.W., Gschwind, M.K., Satterfield, D.L., Sugavanam, K., Coteus, P.W., Heidelberger, P., Blumrich, M.A., Wisniewski, R.W., et al.: The IBM Blue Gene/Q compute chip. Micro, IEEE 32(2), 48–60 (2012), DOI: 10.1109/MM.2011.108

15. Herbordt, M.C., VanCourt, T., Gu, Y., Sukhwani, B., Conti, A., Model, J., DiSabello, D.: Achieving high performance with FPGA-based computing. Computer 40(3), 50 (2007), DOI: 10.1109/MC.2007.79

16. Hoefler, T., Squyres, J.M., Rehm, W., Lumsdaine, A.: A case for nonblocking collective operations. In: In Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops. pp. 155–164. Springer (2006), DOI: 10.1007/11942634_17

17. Kessler, R., Schwarzmeier, J.: Cray T3D: a new dimension for Cray Research. In: Compcon Spring '93, Digest of Papers. pp. 176 –182 (1993), DOI: 10.1109/CMPCON.1993.289660, accessed: 2017-12-19

18. Kumar, S., Dozsa, G., Almasi, G., Heidelberger, P., Chen, D., Giampapa, M.E., Blocksome, M., Faraj, A., Parker, J., Ratterman, J., Smith, B., Archer, C.J.: The deep computing messaging framework: Generalized scalable message passing on the Blue Gene/P Supercomputer. In: Proceedings of the 22nd Annual International Conference on Supercomputing. pp. 94–103. ICS '08, ACM, New York, NY, USA (2008), DOI: 10.1145/1375527.1375544

19. Kumar, S., Mamidala, A., Heidelberger, P., Chen, D., Faraj, D.: Optimization of MPI collective operations on the IBM Blue Gene/Q Supercomputer. Int. J. High Perform. Comput. Appl. 28(4), 450–464 (2014), DOI: 10.1177/1094342014552086

20. Kumar, S., Sabharwal, Y., Garg, R., Heidelberger, P.: Optimization of all-to-all communication on the Blue Gene/L Supercomputer. In: Proceedings of the 2008 37th International Conference on Parallel Processing. pp. 320–329. ICPP '08, IEEE Computer Society, Washington, DC, USA (2008), DOI: 10.1109/ICPP.2008.83

21. Leiserson, C.E., Abuhamdeh, Z.S., Douglas, D.C., et al.: The network architecture of the connection machine CM-5 (extended abstract). In: Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures. pp. 272–285. SPAA '92, ACM, New York, NY, USA (1992), DOI: 10.1145/140901.141883

22. Liu, V.W., Chen, C., Chen, R.B.: Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks. Journal of Combinatorial Optimization 14(2), 131–142 (2007), DOI: 10.1007/s10878-007-9065-5

23. Mai, L., Rupprecht, L., Alim, A., Costa, P., Migliavacca, M., Pietzuch, P., Wolf, A.L.: Netagg: Using middleboxes for application-specific on-path aggregation in data centres. In: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. pp. 249–262. ACM (2014), DOI: 10.1145/2674005.2674996

24. Oak Ridge National Laboratory Leadership Computing Facility: Titan Cray XK7. https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/, accessed: 2017-10-01

25. Ohio State University Network-Based Computing Laboratory: OSU microbenchmarks. http://mvapich.cse.ohio-state.edu/benchmarks/, accessed: 2017-10-01

26. Petrini, F., Coll, S., Frachtemberg, E., Hoisie, A.: Hardware-and-software-based collective communication on the Quadrics network. (2001), http://www.osti.gov/scitech/servlets/purl/975699, accessed: 2017-10-19

27. Russell, R.M.: The CRAY-1 computer system. Commun. ACM 21(1), 63–72 (1978), DOI: 10.1145/359327.359336

28. Schneck, P.B.: Supercomputer Architecture, chap. The CDC STAR-100, pp. 99–117. Springer US, Boston, MA (1987), DOI: 10.1007/978-1-4615-7957-1_5

29. Schneider, T., Hoefler, T., Grant, R., Barrett, B., Brightwell, R.: Protocols for Fully Offloaded Collective Operations on Accelerated Network Adapters. In: Parallel Processing (ICPP), 2013 42nd International Conference on. pp. 593–602 (2013), DOI: 10.1109/ICPP.2013.73

30. Texas Advanced Computing Center: Stampede Supercomputer. https://www.tacc.utexas.edu/systems/stampede, accessed: 2017-12-19

31. Thakur, R., Rabenseifner, R.: Optimization of collective communication operations in MPICH. International Journal of High Performance Computing Applications 19, 49–66 (2005), DOI: 10.1177/1094342005051521

32. Vadhiyar, S.S., Fagg, G.E., Dongarra, J.: Automatically tuned collective communications. In: In Proceedings of SC99: High Performance Networking and Computing. p. 3. IEEE Computer Society (2000), DOI: 10.1109/SC.2000.10024

33. Venkata, M.G., Shamis, P., Sampath, R., Graham, R.L., Ladd, J.S.: Optimizing blocking and nonblocking reduction operations for multicore systems: Hierarchical design and implementation. In: Cluster Computing (CLUSTER), 2013 IEEE International Conference on. pp. 1–8. IEEE (2013), DOI: 10.1109/CLUSTER.2013.6702676

DOI: 10.14529/jsfi170402

Towards Decoupling the Selection of Compression Algorithms from Quality Constraints – An Investigation of Lossy Compression Efficiency

Julian M. Kunkel[1]***, Anastasiia Novikova***[2]***, Eugen Betke***[1]

Data intense scientific domains use data compression to reduce the storage space needed. Lossless data compression preserves information accurately but lossy data compression can achieve much higher compression rates depending on the tolerable error margins. There are many ways of defining precision and to exploit this knowledge, therefore, the field of lossy compression is subject to active research. From the perspective of a scientist, the qualitative definition about the implied loss of data precision should only matter.

With the Scientific Compression Library (SCIL), we are developing a meta-compressor that allows users to define various quantities for acceptable error and expected performance behavior. The library then picks a suitable chain of algorithms yielding the user's requirements, the ongoing work is a preliminary stage for the design of an adaptive selector. This approach is a crucial step towards a scientifically safe use of much-needed lossy data compression, because it disentangles the tasks of determining scientific characteristics of tolerable noise, from the task of determining an optimal compression strategy. Future algorithms can be used without changing application code.

In this paper, we evaluate various lossy compression algorithms for compressing different scientific datasets (Isabel, ECHAM6), and focus on the analysis of synthetically created data that serves as blueprint for many observed datasets. We also briefly describe the available quantities of SCIL to define data precision and introduce two efficient compression algorithms for individual data points. This shows that the best algorithm depends on user settings and data properties.

Keywords: data reduction, compression, lossy, climate data.

Introduction

Climate science is data intense. For this reason, the German Climate Computing Center spends a higher percentage of money on storage compared to computation. While providing a peak compute performance of 3.6 PFLOPs, a shared file system of 54 Petabytes and an archive complex consisting of 70,000 tape slots is provided. Compression offers a chance to increase the provided storage space or to provide virtually the same storage space but with less costs. Analysis has shown that with the proper preconditioning an algorithm can achieve a compression factor of roughly 2.5:1 with lossless compression, i.e., without loss of information [7]. However, the throughput of compressing data with the best available option is rather low (2 MiB/s per core). By using the statistical method in [9] to estimate the actual compression factor that can be achieved on our system, we saw that LZ4fast yield a compression ratio (we define compression ratio as $r = \frac{\text{size compressed}}{\text{size original}}$; inverse is the compression factor) of 0.68 but with a throughput of more than 2 GiB/s on a single core. Therefore, on our system it even outperforms algorithms for optimizing memory utilization such as BLOSC.

Lossy compression can yield a much lower ratio but at expense of information accuracy and precision. Therefore, users have to carefully define the acceptable loss of precision and properties of the remaining data properties. There are several lossy algorithms around that target scientific applications.

[1]Deutsches Klimarechenzentrum, Hamburg, Germany
[2]Universität Hamburg, Hamburg, Germany

However, their definition of the retained information differs: some allow users to define a fixed ratio useful for bandwidth limited networks and visualization; most offer an absolute tolerance and some even relative quantities. The characteristics of the algorithm differs also on input data. For some data, one algorithm yields a better compression ratio than another. Scientists struggle to define the appropriate properties for these algorithms and must change their definition depending on the algorithm decreasing code portability.

In the AIMES project we develop libraries and methods to utilize lossy compression. The SCIL library[3] provides a rich set of user quantities to define from, e.g., HDF5. Once set, the library shall ensure that the defined data quality meets all criteria. Its plugin architecture utilizes existing algorithms and aims to select the best algorithm depending on the user qualities and the data properties.

In the paper [10], we introduced the architecture and idea of the compression library SCIL, together with two new lossy algorithms and analyzed the results for a single data set of a climate model. This paper repeats key concepts from the previous paper but its **key contribution** is providing new experiments and a different perspective by investigating understandable synthetic data patterns in depth. Understanding the general properties (ratio, speed) when compressing different types of data enables us to approximate behavior for similar types of data.

This paper is structured as follows: We give a review over related work in Section 1. The design is described in Section 2. An evaluation of the compression ratios is given in Section 3. Final section provides a summary.

1. Related Work

The related work can be structured into: 1) algorithms for the lossless data compression; 2) algorithms designed for scientific data and the HPC environment; 3) methods to identify necessary data precision and for large-scale evaluation.

Lossless algorithms: The LZ77 [17] algorithm is dictionary-based and uses a "sliding window". The concept behind this algorithm is simple: It scans uncompressed data for two largest windows containing the same data and replaces the second occurrence with a pointer to the first window. DEFLATE is a variation of LZ77 and uses Huffman coding [5]. GZIP is a popular lossless algorithm based on DEFLATE.

Lossy algorithms for floating point data: FPZIP [15] was primarily designed for lossless compression of floating point data. It also supports lossy compression and allows the user to specify the bit precision. The error-bounded compression of ZFP [15] for up to 3 dimensional data is accurate within machine epsilon in lossless mode. The dimensionality is insufficient for the climate scientific data. SZ [4] is a newer and effective HPC data compression method, it uses a predictor and the lossless compression algorithm GZIP. Its compression ratio is at least 2x better than the second-best solution of ZFP. In [7], compression results for the analysis of typical climate data was presented. Within that work, the lossless compression scheme MAFISC with preconditioners was introduced; its compression ratio was compared to that of standard compression tools reducing data 10% more than the second best algorithm. In [6], two lossy compression algorithms (GRIB2, APAX) were evaluated regarding to loss of data precision,

[3]The current version of the library is publicly available under LGPL license:
`https://github.com/JulianKunkel/scil`

compression ratio, and processing time on synthetic and climate dataset. These two algorithms have equivalent compression ratios and depending on the dataset APAX signal quality exceeds GRIB2 and vice versa.

Methods: Application of lossy techniques on scientific datasets was already discussed in [2, 3, 8, 12–14]. The first efforts for determination of appropriate levels of precision for lossy compression method were presented in [1]. By doing statistics across ensembles of runs with full precision or compressed data, it could be determined if the scientific conclusions drawn from these ensembles are similar.

In [9], a statistical method is introduced to predict characteristics (such as proportions of file types and compression ratio) of stored data based on representative samples. It allows file types to be estimated and, e.g., compression ratio by scanning a fraction of the data, thus reducing costs. This method has recently been converted to a tool[4] that can be used to investigate large data sets.

2. Design

The main goal of the compression library SCIL is to provide a framework to compress structured and unstructured data using the best available (lossy) compression algorithms. SCIL offers a user interface for defining the tolerable loss of accuracy and expected performance as various quantities. It supports various data types. In Fig. 1, the data path is illustrated. An application can either use the NetCDF4, HDF5 or the SCIL C interface, directly. SCIL acts as a meta-compressor providing various backends such as the existing algorithms: LZ4, ZFP, FPZIP, and SZ. Based on the defined quantities, their values and the characteristics of the data to compress, the appropriate compression algorithm is chosen[5]. SCIL also comes with a pattern library to generate various relevant synthetic test patterns. Further tools are provided to plot, to add noise or to compress CSV and NetCDF3 files. Internally, support functions simplify the development of new algorithms and the testing.

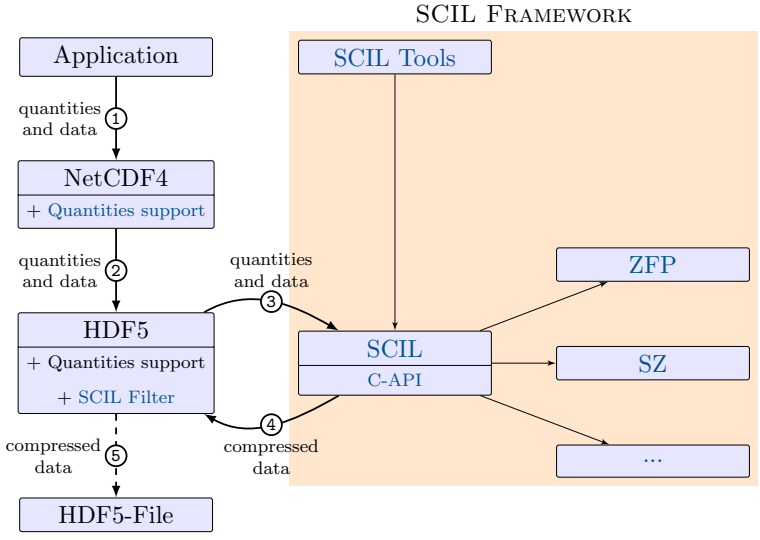


Figure 1. SCIL compression path and components

[4] `https://github.com/JulianKunkel/statistical-file-scanner`

[5] The implementation for the automatic algorithm selection is ongoing and not the focus of this paper. We will model performance and compression ratio for the different algorithms, data properties and user settings.

2.1. Supported Quantities

There are three types of quantities supported:

Accuracy quantities define the tolerable error on lossy compression. When compressing the value v to $\hat{v}$, it bounds the residual error ($r = v - \hat{v}$):

- **absolute tolerance**: $v - \text{abstol} \leq \hat{v} \leq v + \text{abstol}$;
- **relative tolerance**: $v/(1 + reltol) \leq \hat{v} \leq v \cdot (1 + reltol)$;
- **relative error finest tolerance**: used together with rel tolerance; absolute tolerable error for small v's. If relfinest $> |v \cdot (1 \pm reltol)|$, then $v - \text{relfinest} \leq \hat{v} \leq v + \text{relfinest}$;
- **significant digits**: number of significant decimal digits; and
- **significant bits**: number of significant digits in bits.

SCIL must ensure that all the set accuracy quantities are honored, meaning that one can set, e.g., absolute and relative tolerance and the value that is most strict quantity is chosen.

Performance quantities define the expected performance behavior for both compression and decompression (on the same system). The value can be defined according to: 1) absolute throughput in MiB or GiB; or 2) relative to network or storage speed. It is considered to be the expected performance for SCIL but it may not be as strictly handled as the qualities – there may be some cases in which performance is lower. Thus, SCIL must estimate the compression rates for the data, this value is not yet covered by the algorithm selection but will be in the future. The system's performance must be trained for each system using machine learning.

Supplementary quantities: An orthogonal quantity that can be set is the so called *fill value*, a value that scientists use to mark special data points. This value must be preserved accurately and usually is an specific high or low value that may disturb a smooth compression algorithm.

An example for using the low-level C-API is illustrated in Listing 1.

```c
#include <scil.h>
int main(){
  double data[10][20]; // our raw data, we assume it contains sth. useful

  // define the quantities as hints, all specified conditions must hold
  scil_user_hints_t hints;
  hints.relative_tolerance_percent = 10;
  hints.absolute_tolerance = 0.5;
  hints.significant_digits = 2;
  // define performance expectation on decompression speed
  hints.decomp_speed.unit = SCIL_PERFORMANCE_GIB;
  hints.decomp_speed.multiplier = 3.5;
  // ... add more quality constaints if desired
  // create a compression context for a given datatype
  scil_context_t* ctx;
  scil_create_context(&ctx, SCIL_TYPE_DOUBLE, 0, NULL, &hints);

  // the multi-dimensional size of the data, here 10x20
  scil_dims_t dims;
  scil_initialize_dims_2d(& dims, 10, 20);

  // the user is responsible to allocate memory for the output/tmp buffers
  size_t buffer_size = scil_get_compressed_data_size_limit(& dims, SCIL_TYPE_DOUBLE);
  byte * compressed_data = malloc(buffer_size);

  size c_size; // will hold the number of bytes of the compressed buffer
  scil_compress(compressed_data, buffer_size, data, &dims, &c_size, ctx);
  // now do something with the data in compressed_data
```

Listing 1. Usage of the low-level API

2.2. Algorithms

The development of the two algorithms sigbits and abstol has been guided by the definition of the user quantities. Both algorithms aim to pack the number of required bits as tightly as possible into the data buffer. We also consider these algorithms useful baselines when comparing any other algorithm.

2.2.1. Abstol

This algorithm guarantees the defined absolute tolerance. Pseudocode for the Abstol algorithm is provided in Listing 2.

```
1  compress(data, abstol, outData){
2    (min,max) = computeMinMax(data)
3    // quantize the data converting it to integer, according to abstol
4    tmp[i] = round((data[i] - min) * abstol)
5    // compute numbers of mantissa bits needed to store the data
6    bits = ceil(log2(1.0 + (max - min) / abstol))
7    // now pack the neccessary bits from the integers tightly
8    outData = packData(tmp, bits)
9  }
```

Listing 2. Pseudocode for the Abstol algorithm

2.2.2. Sigbits

This algorithm preserves the user-defined number of precision bits from the floating point data but also can honor the relative tolerance. One precision bit means we preserve the floating point's exponent and sign bit as floating point implicitly adds one point of precision. All other precision bits are taken from the mantissa of the floating point data. Notice, that for denormalized number (the leading "hidden" bit is always 0), for example, mantissa 0.0000001 with 5 precision bits will be cutted to 0.00000. That means that the significand part is missed. Note that the sign bit must only be preserved, if it is not constant in the data. Pseudocode for the Sigbits algorithm is illustrated in Listing 3. When a relative tolerance is given it is converted to the number of precision bits.

```
1  compress(data, precisionBits, outData){
2    // preserve the exponent always
3    (sign, min, max) = computeExponentMinMax(data)
4    // compute numbers of bits needed to preserve the data
5    bits = sign + bits for the exponent + precisionBits - 1
6    // convert preserved bits into an integer using bitshift operators
7    tmp[i] = sign | exponent range used | precision Bits
8    // now pack the bits tightly
9    outData = packData(tmp, bits)
10 }
```

Listing 3. Pseudocode for the Sigbits algorithm

Since the values around the 0 contain a huge range of exponents in IEEE floating point representation, e.g., 0 ± 10^{-324} which is usually not needed, the *relative error finest tolerance* limits the precision around 0 to remove exponents and preserve space.

2.3. Compression chain

Internally, SCIL creates a compression chain which can involve several compression algorithms as illustrated in Fig. 2. Based on the basic datatype that is supplied, the initial stage of the chain is entered. Algorithms may be preconditioners to optimize data layout for subsequent

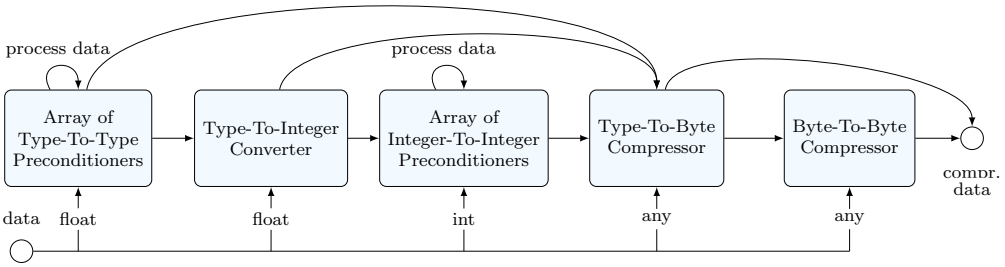


Figure 2. SCIL compression chain. The resulting data path depends on the input data type.

Table 1. Schemas for Pattern Names

Pattern name	Example
random[mutator][repeat]-[min-max]	randomRep10-100
random[min]-[max]	random0-1
steps[number]	steps2
sin[frequency][iterations]	sin35
poly4-[random]-[degree]	poly4-65432-14
simplex[frequency][iterations]	simplex102

If not specified otherwise, min=0 and max=100.

compression algorithms, converters from one data format to another, or, on the final stage, a lossless compressor. Floating point data can be first mapped to integer data and then into a byte stream. Intermediate steps can be skipped.

2.4. Tools

SCIL comes with tools useful for evaluation and analysis: 1) To create synthetic data for compression studies, i.e., well-defined multi-dimensional data patterns of any size; 2) To modify existing data adding a random noise based on the hint set; 3) To compress existing CSV and NetCDF data files.

Synthetic data covers patterns such as constant, random, linear steps (creates a hyperplane in the diagonal), polynomial, sinusoidal or by the OpenSimplex algorithm. OpenSimplex implements a procedural noise generation [11]. An example for the Simplex data is given in Fig. 3; original data and the compressed data for the Sigbits algorithm preserving 3 bits from the mantissa. Each pattern can be parameterized by the min/max value, random seed and two pattern-specific arguments:

- sin: Base frequency (1 means to have one sine wave spanning all dimensions) and number of recursive iterations by which the frequency is doubled each time and the amplitude is halved;
- poly4: Initial number for random number generator and degree of the polynomial;
- steps: Number of steps between min/max; and
- simplex: Initial frequency and number of iterations, in each iteration the frequency is doubled and the amplitude halved.

Additionally, mutators can be applied to these patterns such as step (creating a linear N-dimensional interpolation for the given number of data points) and repeat which repeats a number N-times. Explanation of file names used in this paper are listed in Section 2.4.

3. Evaluation

In the evaluation, we utilize SCIL to compress the data with various algorithms. In all cases, we manually select the algorithm. The test system is an Intel i7-6700 CPU (Skylake) with 4 cores @ 3.40GHz; one core is used for the testing and turbo boost is disabled.

3.1. Test Data

All data uses single precision floating point (32 bit) representation. A pool of data is created 10 times with different random seed numbers from several synthetic patterns generated by SCIL's pattern library and kept in CSV-files. Synthetic data has the dimensionality of (300 x 300 x 100 = 36 MB).

Additionally, we utilize the output of the ECHAM atmospheric model [16] which stored 123 different scientific variables for a single timestep as NetCDF and the output of the hurricane Isabel model which stored 633 variables for a single timestep as binary[6]. The scientific data varies in terms of properties and in particular, the expected data locality. For example, in the Isabel data many variables are between 0 and 0.02 many between -80 and +80 and some are between -5000 and 3000.

3.2. Experiments

For each of the test files, the following setups are run[7]:

- Lossy compression preserving T significant bits
  - Tolerance (T): 3, 6, 9, 15, 20 bits;
  - Algorithms: zfp, sigbits, sigbits+lz4[8];
- Lossy compression with a fixed absolute tolerance
  - Tolerance: 10%, 2%, 1%, 0.2%, 0.1% of the data maximum value[9];
  - Algorithms: zfp, sz, abstol, abstol+lz4.

Each configuration is run 3 times measuring compression and decompression time.

3.3. Understanding Synthetic Patterns

The compression ratios for various synthetic patterns are shown in: Fig. 4 and Fig. 5, the figures show a variable absolute tolerance and the number of precision bits, respectively. Note that for random patterns and simplex patterns, 10 different seeds have been applied and each measurement results in one data point. In most cases, they do not differ notably for a given precision and algorithm.

Absolute tolerance: First, we look at the random patterns in Fig. 4 and the absolute tolerance (left column). The x-axes contains the absolute tolerance between 0.001 and 0.1. In this experiment, the actual tolerance is computed based on the maximum value to enable comparison between different datasets. A value of 0.1 means that 10% of the maximum is used as absolute tolerance. Thus, data can be quantized and encoded in 5 values representing the mean of (0-20%,

[6] `http://vis.computer.org/vis2004contest/data.html`

[7] The versions used are SZ from Aug 14 2017 (git hash 29e3ca1), zfp 0.5.0, LZ4 (Aug 12 2017, 930a692).

[8] This applies first the Sigbits algorithm and then the lossless LZ4 compression.

[9] This is done to allow comparison across variables regardless of their min/max. In practice, a scientist would set the reltol or define the abstol depending on the variable.

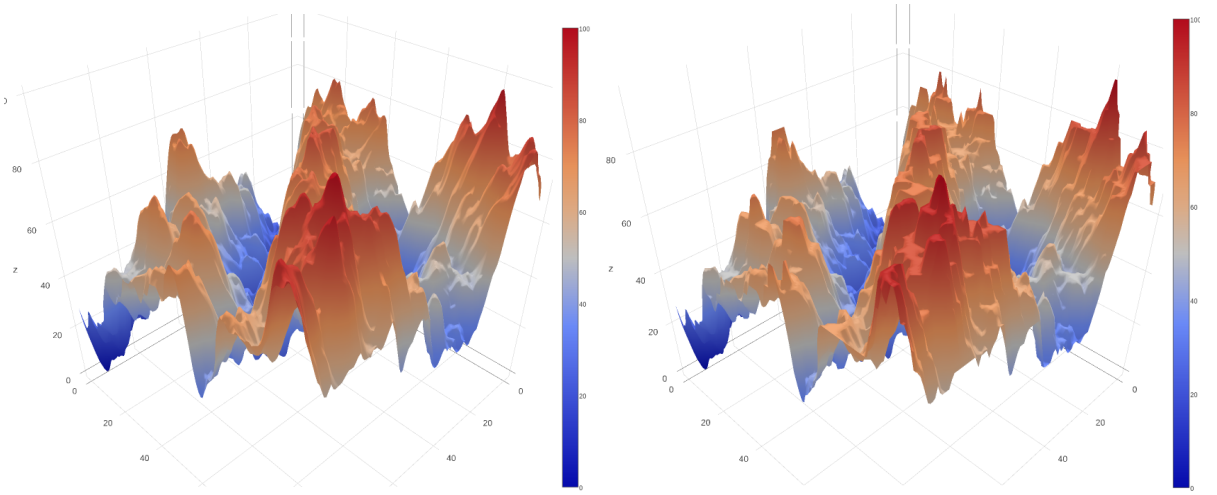


Figure 3. Example of synthetic pattern: Simplex 206 in 2D

20-40%, ...), i.e., 3 bits are needed out of 32 bits ⇒ a compression ratio of 9.4% can be achieved. Therefore, as abstol performs this quantization it is expected to reach that level regardless of the redundancy in the data which it does. ZFP and SZ predict the next data point based on the experience, therefore, as expected the achievable ratio for random data is worse than for abstol by a constant of 0.05. Applying a lossless compressor on pure random data does not help, too. When interpolating between 10x10x10 neighboring cubes (randomIpol10), then the prediction of SZ yields a similar performance to abstol. Abstol+LZ4 improves the ratio slightly for interpolated data as it may reuse some prefix in data. The ratio is a bit better when compressing random data between 1-100 vs. -1 and +1, the reason is that twice the number of intervals are used when data spans a negative and positive range. The randomRep data repeats a value in a block of 2x2x2 or 10x10x10, therefore, SZ and ZFP reduce their ratio but only to 80% and 66% for the small and large block, respectively. This is surprising as 8 and 1000 neighboring points contain the identical value, respectively. For SZ, the interpolation (10x cube) and replication (2x cube) leads to comparable results. Abstol+LZ4 exploits the redundancy better, even better than the GZIP lossy compressor as part of SZ. Note that in most cases the achievable ratio is independent of the random seed for creating the data (ZFP sometimes has some variance).

The polynomial of degree 3 is compressed well by all algorithms, SZ benefits from the locality. Realistic terrain data is created by the simplex algorithm. In Fig. 5, it can be seen that SZ and ZFP can predict this pattern well. Abstol+LZ4 cannot exploit the jitter in the data much but with increasing absolute tolerance, jitter is rounded and repetition can be exploited better. The more refinements are made by the simplex algorithm, the more fine structures are visible, increasing the difficulty for ZFP and Abstol+LZ4 to exploit similarities. The prediction steps of SZ work well in all cases. A similar pattern can be observed with the more simple sinusoidal patterns, they are a bit more uniform and easier to exploit for all algorithms.

Precision bits: Sigbits uses a number of bits needed for encoding the sign bit and exponent range (max - min) and the precision bits as defined by the user. For the random data between 0 to 1, 5 bits are used to represent the exponent[10] leading to a ratio of 22% for 3 precision bits (= 2 mantissa bits). With negative and positive values an extra bit is needed, using interpolation on a range between -1 and +1 may cause additional exponents to emerge, reducing the ratio. Since the remainder of the mantissa looks random, the additional LZ4 stage cannot improve the ratio for

[10]Since the function `rand()` is used to create the test data.

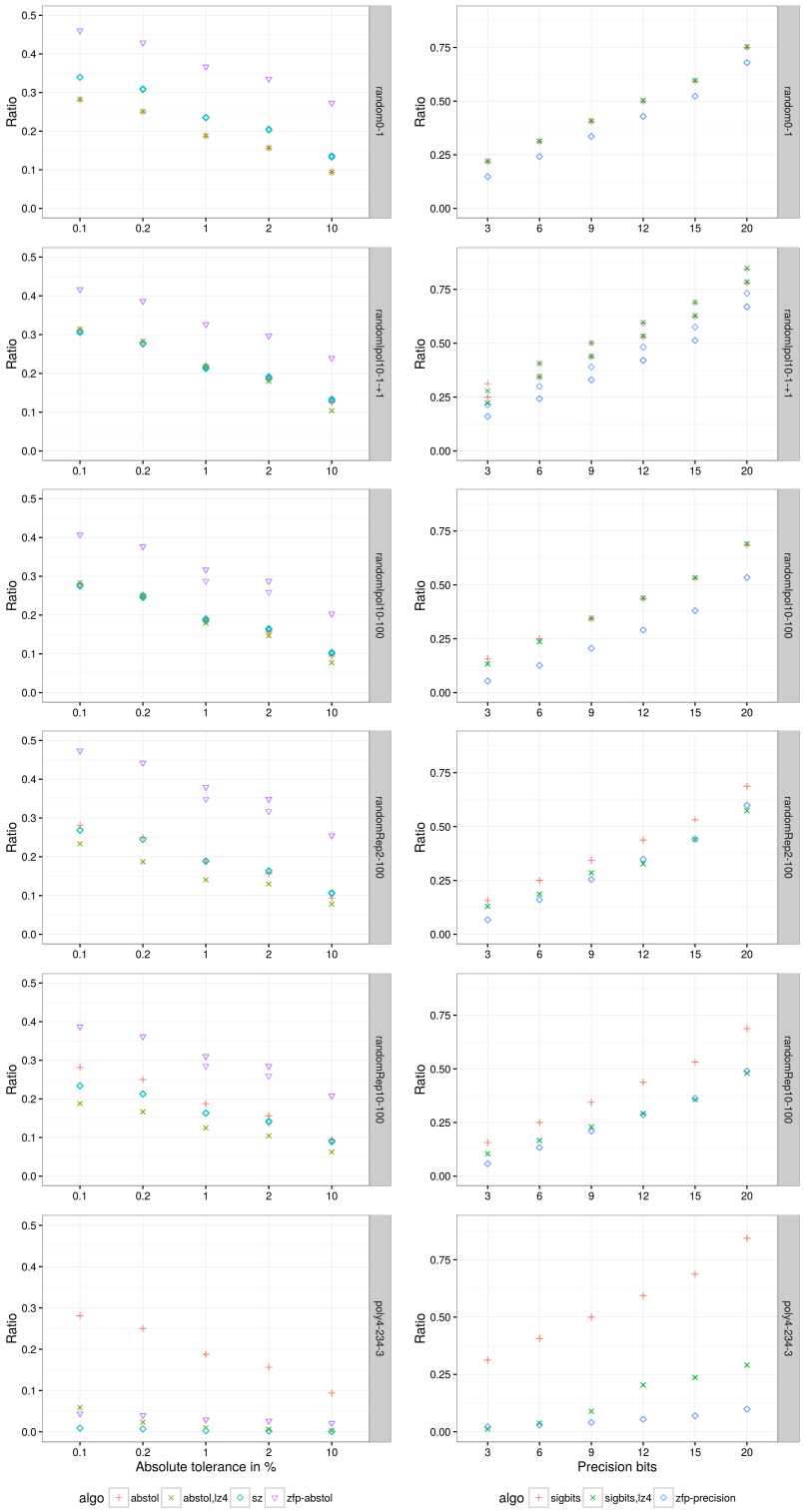


Figure 4. Mean harmonic compression factor for synthetic data based on user settings

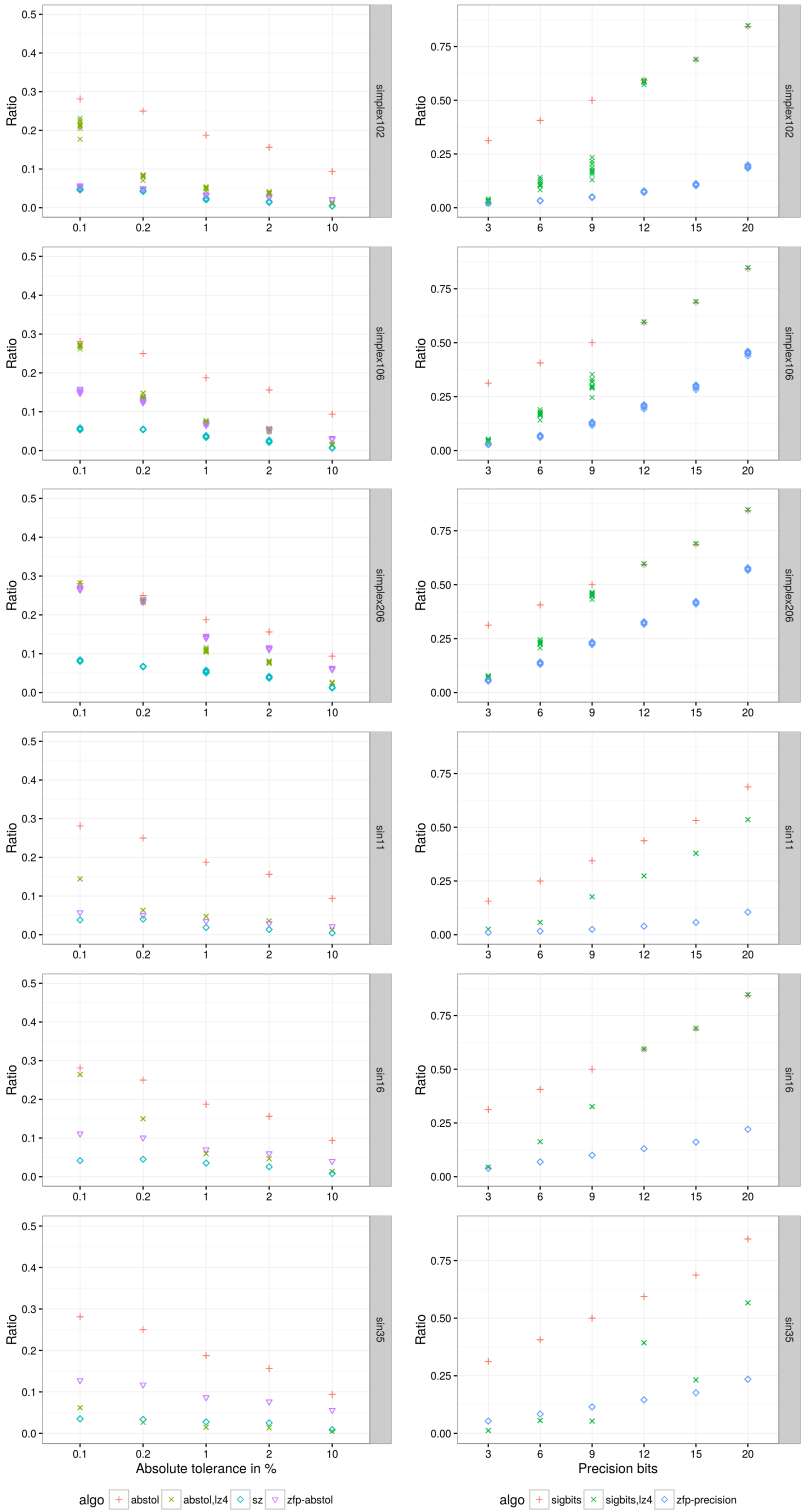


Figure 5. Mean harmonic compression factor for synthetic data based on user settings

interpolated data but only when repeated data blocks emerge. The regular polynomial benefits for a low precision from the repeatability but behaves non-linearly. Similarly, this appears for the variants of the simplex pattern and sin16, the higher the frequency the quicker it degrades. Sin35 behaves non-monotonic for Sigbits+LZ4, with 15 precision bits the ratio is lower than for 12 bits. The reason presumably is that in this case the resulting number of bits is 20 that allows LZ4 to find patterns more effectively (6 bits are needed for the exponent covering the range 1-100, 14 bits for the mantissa).

ZFP cannot be compared easily as it does not support the precision bit quantity but a fixed ratio and compresses data blockwise – this leads to validation errors. For comparison reasons, we use `zfp_stream_set_precision()` with the same number of bits as utilized by Sigbits. Still with a fixed setting, ZFP typically yields a better ratio at the cost of precision.

3.4. Scientific Data

Ratio Depending On Tolerance Next, we investigate the compression factor depending on the tolerance level for the scientific data. The graphs in Fig. 6 and 7 show the mean compression factor for two types of climate data files varying the precision for the algorithms ZFP, SZ, Sigbits and Abstol. The mean is computed on the pool of data, i.e., after compression, a factor of 50:1 means all compressed files occupy only 2% of the original size.

We will discuss the absolute tolerance first: With 1% of tolerance, a compression factor of more than 15 can be achieved on both data sets. For the ECHAM dataset, Abstol+LZ4 outperforms SZ, for the Isabel data SZ outperforms Abstol+LZ4 initially clearly. In both cases, the ratio of Abstol+LZ4 improves with the absolute tolerance.

When using precision bits, ZFP yields a better factor on the Isabel data than Sigbits. However, note that since this dataset uses high fill values, the result cannot be trusted. With 3 precision bits (relative error about 12.5%), a ratio of around 10 is achievable. It can be concluded that the ECHAM dataset is more random compared to the Isabel data which stores a finer continuous grid.

Fixed Absolute Tolerance To analyze throughput and compression ratio across variables, we selected an absolute tolerance of 1% of the maximum value.

Mean values are shown in Tbl. 2a. The synthetic random patterns serve as baseline to understand the benefit of the lossy compression; we provide the means for the different random patterns. Abstol+LZ4 yields a 20% reduction compared to SZ for ECHAM and the random data while for Isabel data it needs about 50% more space. Compression and decompression

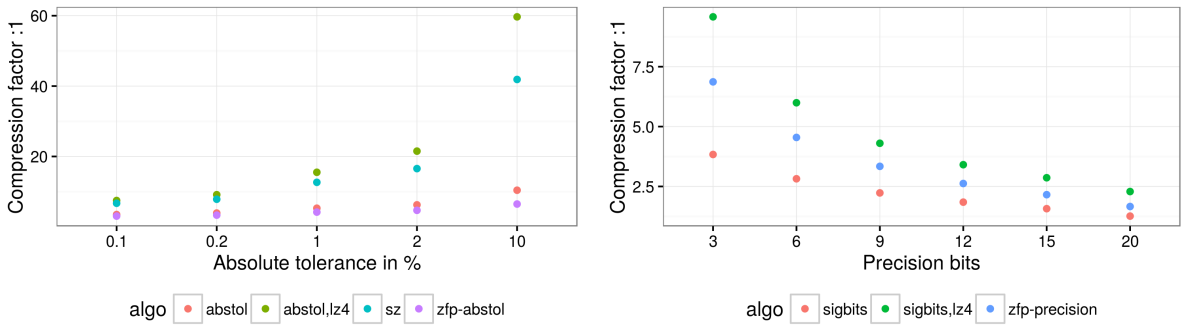


Figure 6. Mean harmonic compression factor for ECHAM data based on user settings

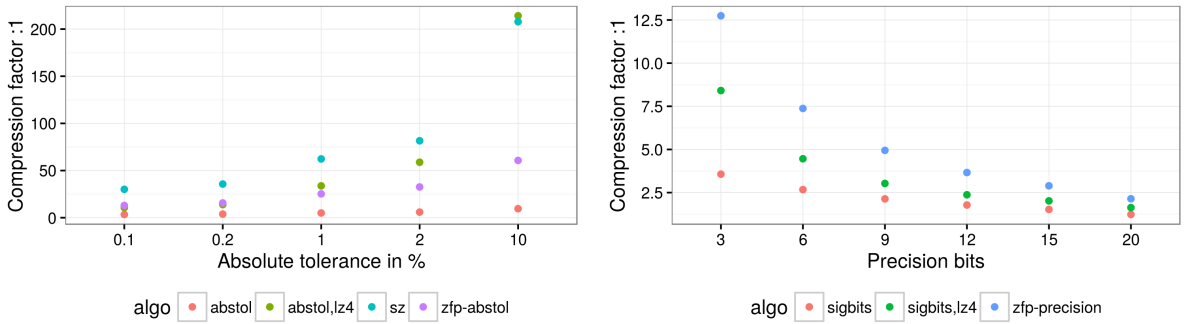


Figure 7. Mean harmonic compression factor for Isabel data based on user settings

speed of Abstol+LZ4 is at least 2x the speed of SZ. LZ4 supports the detection of random data and avoids compression in that case increasing performance for random data significantly. ZFP achieves worse compression ratios but with a better compression speed than SZ. For Isabel data the decompression is even a bit higher than when using Abstol+LZ4.

The results for the individual climate variables are shown in Fig. 8 for ECHAM and Isabel data; on the x-axis are the different variables sorted on compression ratio to ease identification of patterns, e.g., to see the impact of higher compression ratio to performance. It can be observed that for the ECHAM data Abstol+LZ4 yields in most cases the best compression ratio and the best compression and decompression speeds. For some variables (on the right), SZ compresses better. When dealing with Isabel data, in most cases SZ outperforms the ratio of Abstol+LZ4, but for a few Abstol+LZ4 is better. SZ performance is quite robust and so is Abstol but the LZ4 step depends on the compressability on the data. Both Abstol+LZ4 and SZ reveal some steps in the data, where the complexity to compress data increases.

Table 2. Harmonic Mean Compression of Scientific Data

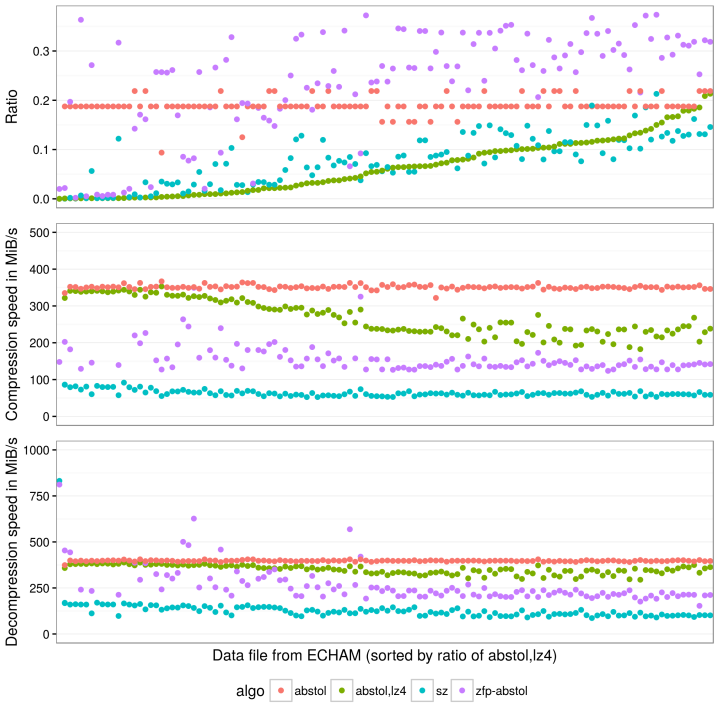
	Algorithm	Ratio	Compr. MiB/s	Decomp. MiB/s
ECHAM	abstol	0.190	260	456
	abstol,lz4	0.062	196	400
	sz	0.078	81	169
	zfp-abstol	0.239	185	301
Isabel	abstol	0.190	352	403
	abstol,lz4	0.029	279	356
	sz	0.016	70	187
	zfp-abstol	0.039	239	428
Random	abstol	0.190	365	382
	abstol,lz4	0.194	356	382
	sz	0.242	54	125
	zfp-abstol	0.355	145	241

a) 1% absolute tolerance

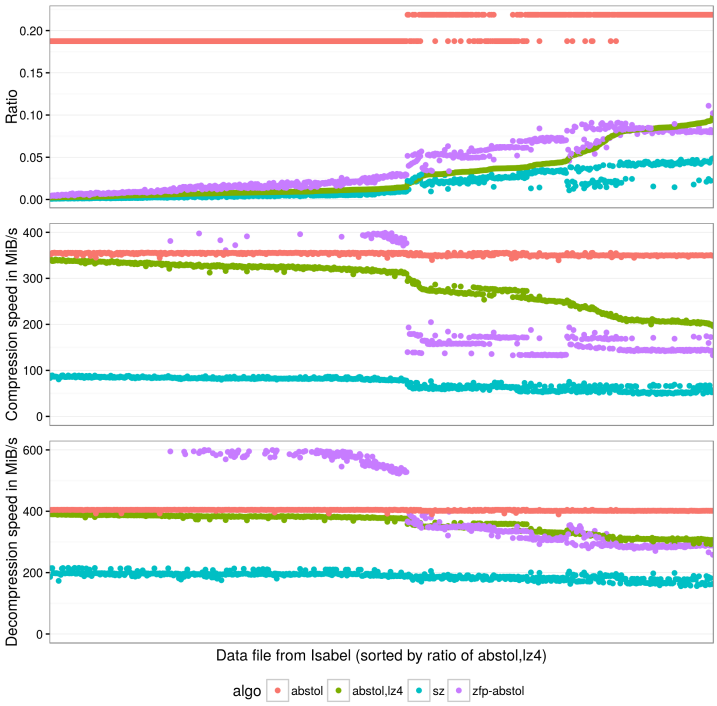
	Algorithm	Ratio	Compr. MiB/s	Decomp. MiB/s
ECHAM	sigbits	0.448	462	615
	sigbits,lz4	0.228	227	479
	zfp-precision	0.299	155	252
Isabel	sigbits	0.467	301	506
	sigbits,lz4	0.329	197	366
	zfp-precision	0.202	133	281
Random	sigbits	0.346	358	511
	sigbits,lz4	0.348	346	459
	zfp-precision	0.252	151	251

b) 9 bits precision

Fixed Precision Bits Similarly to our previous experiment, we now aim to preserve 9 precision bits. mean values are given in Tbl. 2b and Fig. 9 shows the ratio and performance across



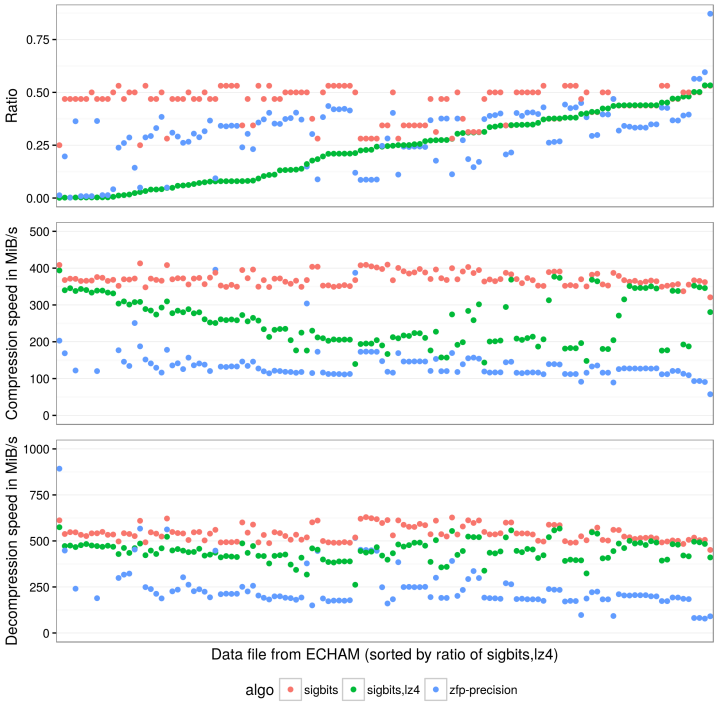
a) ECHAM



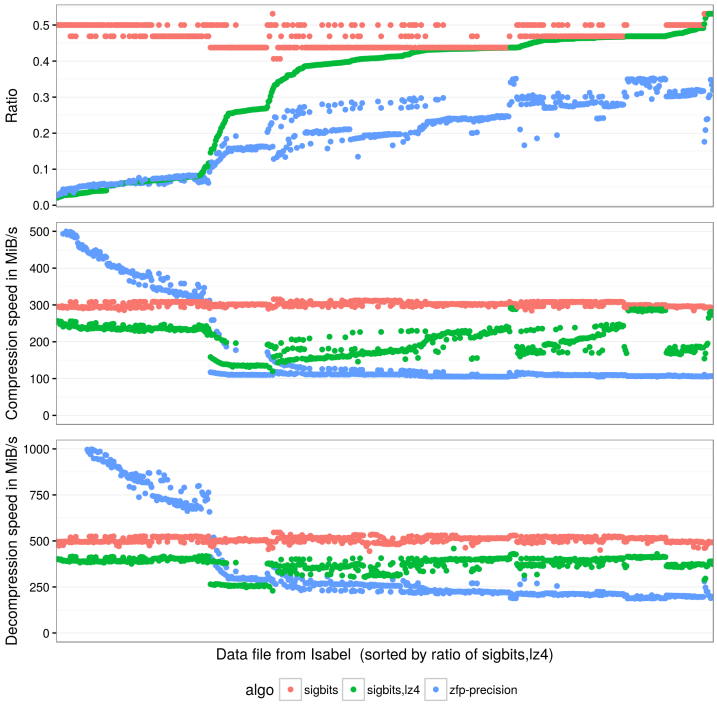
b) Isabel

Figure 8. Compressing climate data variables with an absolute tolerance of 1% max

climate variables. The Sigbits algorithm is generally a bit faster than Abstol. It can be seen that Sigbits+LZ4 outperforms ZFP mostly for ECHAM data, although ZFP does typically not hold the defined tolerance. For Isabel data ZFP is typically better than Sigbits+LZ4 (but again does not hold the precision bits). In this case, the LZ4 compression step is beneficial for a small fraction of files, for 50% it does not bring any benefit.



a) ECHAM



b) Isabel

Figure 9. Compressing climate data variables with 9 precision bits

Summary

This paper describes the concepts for the scientific compression library (SCIL) and compares algorithms with the state-of-the-art compressors ZFP and SZ. We investigate various relevant synthetic test patterns in more detail. As expected the predictors of SZ and ZFP improve the ratio for continuous data, for data containing repeated patterns or with more randomness, Abstol can yield a comparable or better ratio. The structure of the data depends on the scientific meaning of the variable and the scientist setting up the experiment: the output may be a continuous high resolution variable that is highly predictable or a downsampled version, it may contain not differentiable data potentially mixed with extreme values (e.g., fill values that mask points). The difference is relevant since the pattern that repeats a single value within 3D data block, the compression ratios of all algorithms are behind the expectation. Instead of storing 10x10x10 blocks of the same value, a single point could be stored for each block allowing to reduce the data volume to 1/1000 while the best algorithm here only yields a reduction to 1/33.

In any case, the performance of Sigbits and Abstol are better than existing algorithms. Since SCIL aims to choose the best algorithm automatically, it ultimately should be able to take benefit of both algorithms. We believe that decoupling of the interface to define scientific precision from the actual algorithm is a mandatory step for the HPC community to move on. Ongoing work is the development of an algorithm honoring all quantities at the same time and the automatic chooser for the best algorithm.

Acknowledgements

This work was supported in part by the German Research Foundation (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) (GZ: LU 1353/11-1). Thanks for the Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the U.S. National Science Foundation (NSF).

References

1. Baker, A.H., Hammerling, D.M., Mickelson, S.A., Xu, H., Stolpe, M.B., Naveau, P., Sanderson, B., Ebert-Uphoff, I., Samarasinghe, S., De Simone, F., Gencarelli, C.N., Dennis, J.M., Kay, J.E., Lindstrom, P.: Evaluating lossy data compression on climate simulation data within a large ensemble. Geoscientific Model Development, 9 pp. 4381–4403 (2016), DOI: 10.5194/gmd-9-4381-2016

2. Bautista-Gomez, L.A., Cappello, F.: Improving floating point compression through binary masks. In: Hu, X., Lin, T.Y., Raghavan, V.V., Wah, B.W., Baeza-Yates, R.A., Fox, G.C., Shahabi, C., Smith, M., Yang, Q., Ghani, R., Fan, W., Lempel, R., Nambiar, R. (eds.) Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA. pp. 326–331. IEEE (2013), DOI: 10.1109/BigData.2013.6691591

3. Bicer, T., Agrawal, G.: A Compression Framework for Multidimensional Scientific Datasets.

Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), 2013 IEEE 27th International pp. 2250–2253 (2013), DOI: 10.1109/IPDPSW.2013.186

4. Di, S., Cappello, F.: Fast Error-bounded Lossy HPC Data Compression with SZ. In: Parallel and Distributed Processing Symposium, 2016 IEEE International. pp. 730–739. IEEE (2016), DOI: 10.1109/IPDPS.2016.11

5. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proceedings of the IRE 40(9), 1098–1101 (1952), DOI: 10.1109/JRPROC.1952.273898

6. Hübbe, N., Wegener, A., Kunkel, J., Ling, Y., Ludwig, T.: Evaluating Lossy Compression on Climate Data. In: Kunkel, J.M., Ludwig, T., Meuer, H.W. (eds.) Supercomputing. pp. 343–356. No. 7905 in Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2013), DOI: 10.1007/978-3-642-38750-0_26

7. Hübbe, N., Kunkel, J.: Reducing the HPC-Datastorage Footprint with MAFISC – Multidimensional Adaptive Filtering Improved Scientific data Compression. Computer Science - Research and Development pp. 231–239 (2013), DOI: 10.1007/s00450-012-0222-4

8. Iverson, J., Kamath, C., Karypis, G.: Fast and effective lossy compression algorithms for scientific datasets, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7484, pp. 843–856. Springer (2012), DOI: 10.1007/978-3-642-32820-6_83

9. Kunkel, J.: Analyzing Data Properties using Statistical Sampling Techniques – Illustrated on Scientific File Formats and Compression Features. In: Taufer, M., Mohr, B., Kunkel, J. (eds.) High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P3MA, VHPC, WOPSSS. pp. 130–141. No. 9945 2016 in Lecture Notes in Computer Science, Springer (2016), DOI: 10.1007/978-3-319-46079-6_10

10. Kunkel, J., Novikova, A., Betke, E., Schaare, A.: Toward Decoupling the Selection of Compression Algorithms from Quality Constraints. In: High Performance Computing. No. 10524 in Lecture Notes in Computer Science, Springer (2017), DOI: 10.1007/978-3-319-67630-2_1

11. Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D.S., Lewis, J.P., Perlin, K., Zwicker, M.: A survey of procedural noise functions. In: Computer Graphics Forum. vol. 29, pp. 2579–2600. Wiley Online Library (2010), DOI: 10.1111/j.1467-8659.2010.01827.x

12. Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., Samatova, N.: Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-Temporal Data. European Conference on Parallel and Distributed Computing (Euro-Par), Bordeaux, France (2011), DOI: 10.1007/978-3-642-23400-2_34

13. Laney, D., Langer, S., Weber, C., Lindstrom, P., Wegener, A.: Assessing the Effects of Data Compression in Simulations Using Physically Motivated Metrics. Super Computing (2013), DOI: 10.3233/SPR-140386

14. Lindstrom, P.: Fixed-Rate Compressed Floating-Point Arrays. IEEE Transactions on Visualization and Computer Graphics 2012 (2014), DOI: 10.1109/BigData.2013.6691591

15. Lindstrom, P., Isenburg, M.: Fast and efficient compression of floating-point data. IEEE transactions on visualization and computer graphics 12(5), 1245–1250 (2006), DOI: 10.1109/TVCG.2006.143

16. Roeckner, E., Bäuml, G., Bonaventura, L., Brokopf, R., Esch, M., Giorgetta, M., Hagemann, S., Kirchner, I., Kornblueh, L., Manzini, E., et al.: The atmospheric general circulation model ECHAM 5. PART I: Model description (2003)

17. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on information theory 23(3), 337–343 (1977), DOI: 10.1109/TIT.1977.1055714

DOI: 10.14529/jsfi170403

Adaptive Load Balancing Dashboard in Dynamic Distributed Systems

Seyedeh Leili Mirtaheri[1], *Seyed Arman Fatemi*[2], *Lucio Grandinetti*[3]

Considering the dynamic nature of new generation scientific problems, load balancing is a necessity to manage the load in an efficient manner. Load balancing systems are used to optimize the resource consumption, maximize the throughput, minimize response time, and to prevent overload in resources. In current research, we consider operational distributed systems with dynamic variables caused by different nature of the applications and heterogeneity of the various levels in the system. The conducted studies indicate that many different factors should be considered to select the load balancing algorithm, including the processing power, load transfer and communication delay of nodes. In this work, We aim to design a dashboard that is capable of merging the load balancing algorithms in different environments. We design an adaptive system infrastructure with the ability to adjust various factors in the run time of a load balancing algorithm. We propose a task and a resource allocation mechanism and further introduce a mathematical model of load balancing process in the system. We calculate a normalized hardware score that determines the maturity of the system according to the environmental conditions of the load balancing process. The evaluation results confirm that the proposed method performs well and reduces the probability of system failure.

Keywords: Load Balancing, Resource Management, Dynamic Variables, Communication Delay, Distributed System.

Introduction

In the next-generation high performance computing (HPC) systems, the scientific programs are going to get more complex with unpredictable requests what requires large scale and more powerful computing systems [3]. Distributed computing system (DCS) is a promising solution to such demands. DCS consists of a group of computers, each of which has an independent operating system and all are connected through a network [12]. This definition of distributed computing systems is very general and covers all types of purposes for user resource connectivity. Users get connected to resources to receive different facilities and services like computing, file sharing, I/O sharing, and storage. In distributed computing systems with mission of high performance computing, the main concern of managing resources is reaching to minimum execution time and maximum throughput in running scientific/industrial programs [10].

In resource management framework, the load balancer is the key unit in efficient utilization of resources and improving response time [16]. Load balancing is the distribution of load among several computing resources such as computers, clusters, network communication lines, central processing units, or disk drives. The advantages of the load balancing could be categorized as: optimizing resource consumption, maximizing network throughput, reducing response time, and preventing overload in each resource. It is known that using multiple components in load balancing increases reliability and availability [5].

In load balancing mechanism, some parameters have direct impact in efficiency of load balancing operation. A group of these parameters are determined by specifying the status of nodes [1]. Practically, the nodes speed up in executing the commands and their current load are

[1]Department of Electrical & Computer Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran
[2]Kharazmi University, Tehran, Iran
[3]Department of Electronics, Informatics, and Systems, University of Calabria (Unical), Rende, Italy

two important factors in specifying the status of the nodes and making decision to distribute the load based on them. It is clear that, the load balancer should migrate and transfer the extra load from over loaded nodes to the ones that are under loaded. Another group of parameters are related to execution of load balancing mechanism that among them specifying the start time of load balancing mechanism is an important issue. In one-shot strategy, when extra load is imposed in the node, the load balancing operation starts to work [11]. The third group of parameters are related to interconnection platform status. The important effective parameter in load balancing mechanism from these parameters is the communication delay between the nodes.

Load balancing should not only be considered in terms of a mechanism that can improve the computing efficiency. Load balancing is a necessity for large scale computing systems. In such systems, the whole computation goal may fail if the load balancing is not applied in the system. on the other-hand, predetermined variables are commonly used in decision-making algorithms by introducing unchangeable formulas and computing algorithms. Therefore, beside efficient and optimized algorithms for load balancing, it is important to introduce a framework dashboard that can also adapt to variation of parameters by choosing different algorithms, dynamically. Therefore, the two usage of load balancing systems are as follows:

1. Reducing critical points in system: In fact, using load balancing improves the system availability. If an adaptive load balancing system is used in the network and a node (hardware or software) corrupts, no failure time is felt from the user's side.
2. Efficient load distribution by load balancing exploits the full computational power (maximum available) of system elements.

Standard load balancing manager includes five activities: queuing, scheduling, monitoring, resource control and accountants. As shown in Fig. 1, queuing is the first phase of load balancing operation and in this unit the tasks are assigned by user, and the process ends by giving the result of executed tasks to the user. The aim of these five activities is efficient matching of user-specified workload to existing resources by keeping balance the load of the system. The standard process of load balancing manager starts by receiving the jobs from users in queuing unit. After placement of jobs in queues, scheduling unit is activated. Scheduling is a fundamental technique for improving performance in computer systems. The scheduler has to balance the priorities of each job, with the demands of other jobs, the existing system compute and storage resources, and the governing policies dictated for their use by system administrators. Scheduler should be able to deal with different types of jobs such as huge jobs, small jobs, real-time jobs, high-priority jobs, dynamic jobs, static jobs and etc. The scheduler governs the order of execution based on these independent priority assessments.

The present study aims to develop a load balancing dashboard to monitor the impact of static and dynamic variables on the non-stop decision making process by the load balancer. In current research, we consider distributed systems with dynamic variables. We aim to design and model an adaptive dashboard that is capable to merge the load balancing algorithms in different environments. We design an adaptive system infrastructure with the ability to adjust various factors in the run time of a load balancing algorithm. We propose a task and a resource allocation mechanism and further introduce a mathematical model of load balancing process in the system. We calculate a normalized hardware score that determines the maturity of system according to the environmental conditions of the load balancing process. The rest of this paper is organized as follows: In section 1, we present the related works based on some recent researches

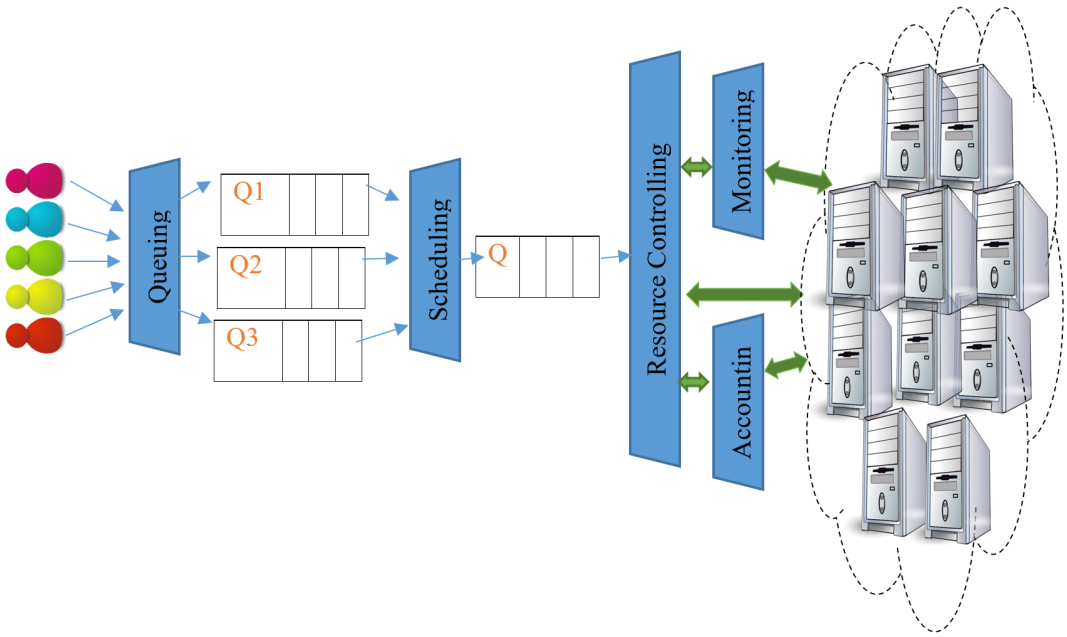


Figure 1. Standard load balancing manager scheme

and the investigation of Azure Load Balancing system. The effect of communication delay on load balancing is presented in Section 2 and Section 3 includes our proposed task and resource allocation mechanism. The mathematical model and scoring model are presented in Section 4, and evaluation results of the proposed method are presented in Section 5. Finally, Section 6 concludes the paper.

1. Related Works

1.1. Related Researches

Research studies conducted in this area pursued specific goals. The summarized significant goals in the literature are as follows:

1. In some of the studies, the goal is to propose new methods to improve the performance of current algorithms for different environments. These studies investigate the performance of the existing algorithms and proposed combined algorithms derived from the existing ones [15]. Usually, the authors compare the response time of the proposed algorithms with the existing ones by simulating the environment through simulation tools.
2. Some studies focus on load balancing algorithms in particular environments (such as public clouds) [19]. Generally, this approach aims to adapt and optimize load balancing algorithms based on the implementation variables.
3. Some researchers focus on mathematical approaches to improve the performance of load balancing algorithms. For example, heuristic optimization and artificial intelligence techniques [8].
4. Some studies focus on the scalability [13]. The scalability is related to the implementation structure. Normally, the workload of load balancing structures is not predictable (for example, when a load balancing structure is needed for dynamic load distribution among processors). Accordingly, the scalability may result in an adaptive load balancing system for a larger number of environments.

One of the main challenges of performing a load balancing process, is the right choice of algorithm. This choice, as mentioned earlier, is influenced by many factors. A lot of studies were conducted to find a proper algorithm. In [9] authors propose a dynamic parallel scheduling algorithm for load balancing. Several strategies are proposed to distribute the load using dynamic and distributed load balancing mechanisms, called sender-initiated and receiver-initiated strategies. In the sender-initiated strategy, a portion of the load of over-loaded processor is sent to another computing node [2]. In receiver-initiated strategy, the under-loaded processor initiates the transaction by sending a message to other nodes [6]. In [7], authors propose a load balancing mechanism for massive parallel computations based on sparse grid combination techniques. In [18], authors present an optimized load balancing algorithm by organizing the tasks in queues based on task data size and location. The implemented technique is a MATRIX. However, the portion of the load that should be transferred by forming unpredictable events in runtime is not specified. The proposed method in [17], named Slum++, employs multiple controllers so that each one manages a partition of computing nodes and participates in resource allocation through resource balancing techniques. The authors propose a monitoring-based weakly consistent resource stealing technique to achieve resource balancing in distributed high performance computing. In [14], authors investigate migration of applications for high performance computing by deriving respective requirements for specific field of applications. In [10], authors consider practical parameters such as communication delay and propose an optimized dynamic load balancing algorithm for exasclae distributed computing systems.

1.2. Related Measures

1.2.1. Azure Load Balancing

In this section we review the Azure Load Balancing (ALB) system as a successful and practical business case as illustrated in Fig. 2. The ALB system is a load balancing service based on the process as a Service, which provides the highest level of network performance as well as the highest level of availability under UDP and TCP protocols. This system splits the traffic among healthy devices in a cloud structure or among distributed virtual machines. ALB is commonly used in the following cases:

- Splitting traffic of Internet access among virtual machines;
- Splitting traffic among virtual machines in a virtual network, among virtual machines in a cloud service, or among virtual machines and local computers in a local interconnect network; and
- Conducting external traffic on a specific virtual machine.

1.2.2. Required Attributes of ALB

In this section, we address the required information to register a task based on a practical sample of Azure Load Balancing. The following information is collected from the official website of the Ohio Super Cloud Computing Center. This website uses a batch file to allocate tasks to servers. In fact, this is a text file containing task allocation settings. The proprietary file formats are .job or .pbs. This file has its own structure and supports features like commenting. This center is used to accept a specific scripting task. The required attributes to register a task are as follows:

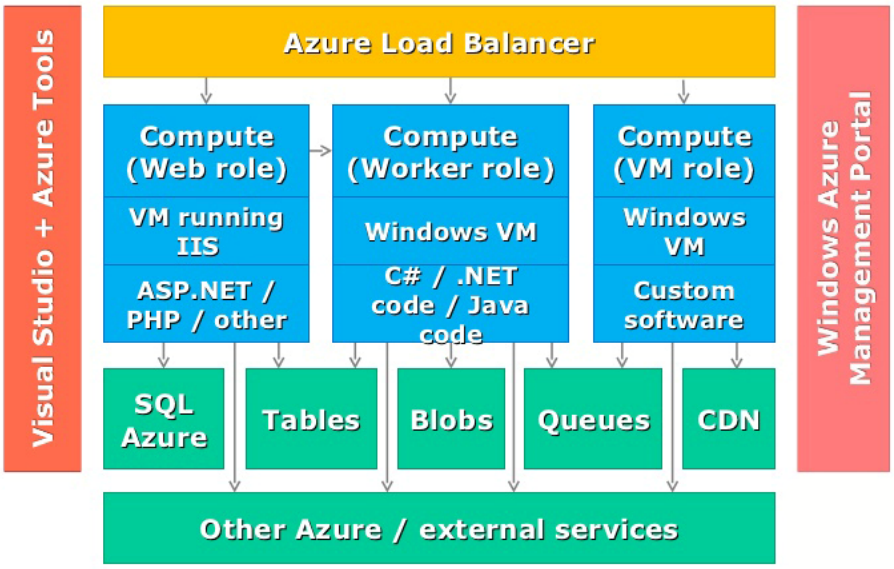


Figure 2. General architecture of Azure Load Balancing

1. **Wall Back Time**: It represents the maximum allowed runtime for Job. This attribute is accepted in two formats: second and HH: MM: SS. This is an estimated time, and if our work exceeds it, it will be eliminated by the system.
2. **Nodes**: Through this attribute, the number of nodes and node characteristics are specified. It is also used to specify the number of processors per node or PPN, the number of GPUs per node or GPUN, and the type of node.
3. **Memory**: This attribute determines the total amount of memory in all nodes. This feature is used in special cases when requiring a high-memory node, or when there is no matching between the number of processors in the node and requested memory.

2. Effect of Communication Delay on Load Balancing

Delay time of load transferring from the nodes with positive load (over-loaded nodes) to nodes with negative load (under-loaded nodes) should be acceptable and reasonable. If this amount of the transferring delay be more than the execution time of instructions in that node, load balancer should use suitable policy such that the total time of program execution be efficient. For example, consider a node that includes 200 task for execution and the average number of task execution in this node is 100 tasks per second. In this situation, the load balancer of the system makes decision to transfer 80 tasks to other node, therefore this node should execute 120 task that due to execution time of one task in this node the total time executing 120 task will be 1.2 task/s. Now suppose that transferring time of each task to destination node is 0.02 second. In this situation, for transferring 80 tasks, 1.6 seconds is needed. Therefore, based on this load balancing policy, the source node (the node that distribute its extra load) dont have any task for execution at least for 0.4 second. On the other hand, the destination node also can finish its tasks execution long before receiving these new tasks (transferred load). In both cases, part of the capacity of the system that could be used to execute commands will be lost. This communication delay is considered in [10]. In this research authors introduce a compensating factor and find the optimum value based on the modeled optimization problem.

To illustrate the effect of load transfer delay, we consider a scenario and assume that all nodes have 300 initial tasks. The load execution speed is considered to be 200 tasks per second for each node and 100 tasks per second are transferred from one node to another. In this scenario, we assume that the external load is inserted to node 6 having node 4 as its sole neighbor depicted in Fig. 3.

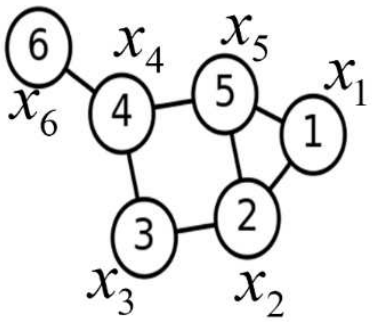


Figure 3. Graph of a sample distributed system model

As depicted in Fig. 4, the node 4 performs all assigned tasks at t=0.17 and is in standby state till t=5.05 when the load from load balancing is reached to node 4. The load balancing is performed in a way that all nodes finish their tasks together. The standby time of node 4 causes extra delay in the system and node 4 finish its assigned tasks at t=7.13 while node 6 ends the tasks at t=2.24. According to the definition of total exertion time, the network execution time for assigned tasks is equal to t=7.13. So, it is obvious to compensate for this load transfer delay and to prevent the standby time in the system [10].

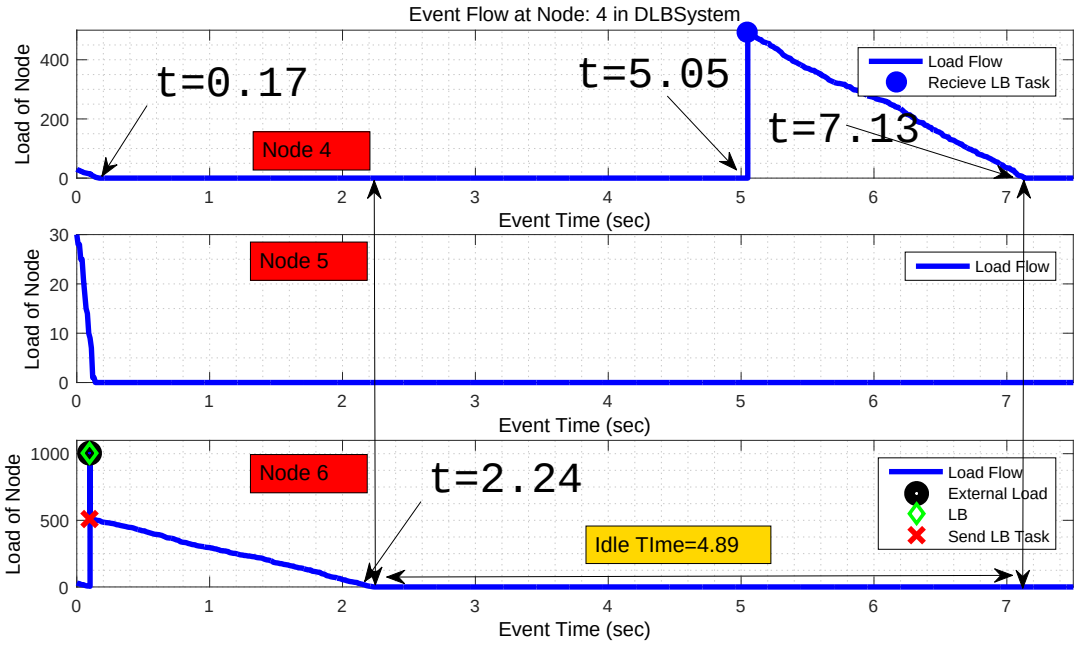


Figure 4. Effect of load transfer delay on idle time of systems

3. Proposed Mechanism

In this section, we outline the proposed system. The innovation of proposed mechanism is described in the following two subsections:

- Task allocation process; and
- Resource allocation process

The main procedure is depicted in Fig. 5. First the user needs to specify the resource allocation process to prioritize tasks for execution after determining the intended user algorithm (without the need for coding unlike the existing simulators). In the final step, the user executes the load balancing procedure and views the results.

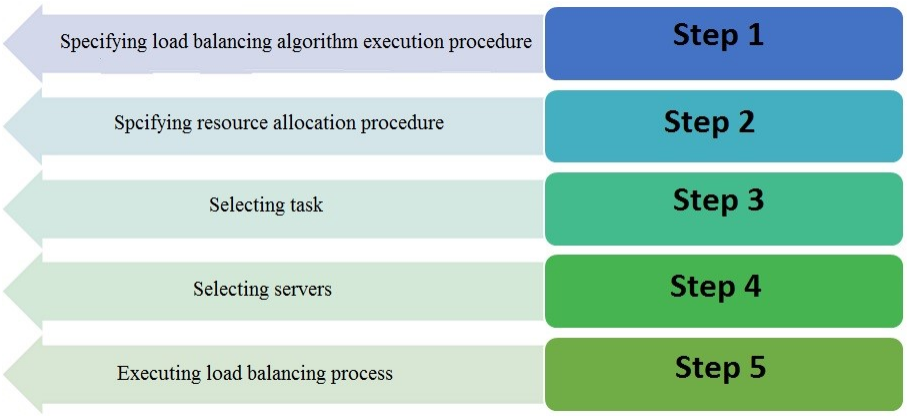


Figure 5. The general procedures of load balancing in the proposed system

3.1. Entities Participating in the Load Balancing Process

3.1.1. Virtual Servers

The structure of the entity in the database is depicted in Fig. 6. In simple words, each virtual server can have one or more Hardware in this system. Each Hardware has exactly one Hardware Type, and each Hardware Type has one Unit. This structure is used to explain how the load balancing algorithm is implemented.

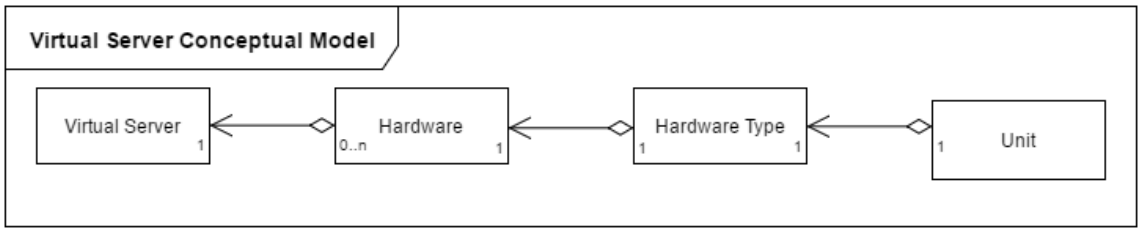


Figure 6. Conceptual model of virtual servers

3.1.2. Job/Task

The structure of task/job in the database is depicted in Fig. 7. Each job can have one or more Hardware Type in this system, and each Hardware Type in the Job needs a consumption. Therefore, we specify the hardware requirements for different hardware defined in system (in a

completely dynamic manner). We also use this structure to explain how to implement the load balancing algorithm in the following section.

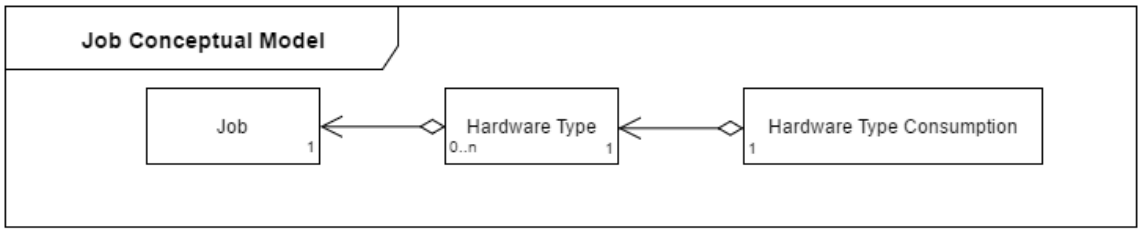


Figure 7. Conceptual model of job/task

3.2. Resource Allocation Process

The proposed system can facilitate implementing the required algorithms and allocating resources in the environment. The general approach is to enable the user to implement algorithms and allocate resources without dealing with complex issues (compared to other existing simulators, such as SimGrid [4]), by using the structures described in the following section.

3.2.1. Load Balancing and Resource Allocation Process

The load balancing and resource allocation processes are implemented in two ways that are described separately.

Load balancing process using predetermined algorithms: In this model, the load balancing and resource allocation processes are performed using the following steps:

1. Selecting tasks (jobs) from the task list defined in the system.
2. Selecting servers from the server list defined in the system.
3. Selecting an algorithm from the following algorithms for selecting tasks.
4. Applying the settings for the resource allocation process.
5. Executing the load balancing process using predefined formulas.

In this procedure, the load balancing process is executed according to the selected settings for all available queues of job. However, first, the tasks need to be sorted according to the selected algorithm to form queues. Simply put, the load balancing process is fully executed for all the job queues. After a complete execution, the load balancing process is executed for the next task in the queue for all servers (available in the server queue) according to the server settings. This procedure is summarized in Fig. 8.

Load balancing process using dynamic algorithms: The objective is to create a dynamic environment for executing load balancing algorithms according to the user requirements. The proposed system provides the ability to implement different load balancing algorithms in runtime through different settings applied by user. This model is depicted in Fig. 9.

4. Mathematical Model of the Proposed System

The load balancing process in proposed system, includes the following steps:

Selecting the task with the highest priority: The task with highest priority is selected from the queue and is applied to the available servers in the queue to start the load balancing process. **Filter 1 on servers**: This filter removes all servers that do not meet all the hardware

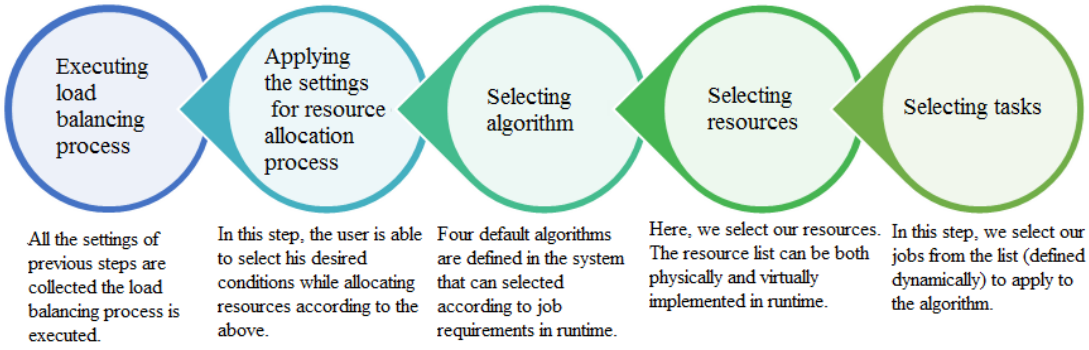


Figure 8. Load balancing process using predetermined algorithms

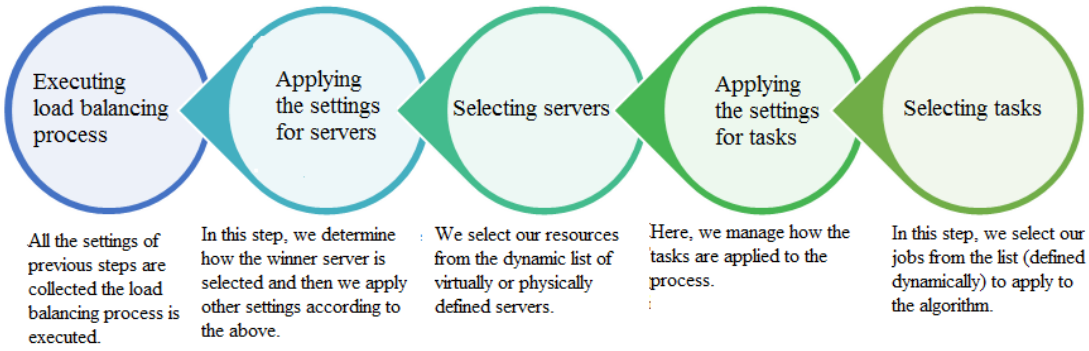


Figure 9. Load balancing process using dynamic algorithms

requirements. For example, if a job requires four different types of hardware, only the servers that meet these requirements remain in the list.

$$\textbf{virtual Server} \Rightarrow$$
$$\textbf{Distinct(HardwareType in Jobs)} \in \textbf{Distinct(HardwareType in VirtualServere)} \quad (1)$$

Filter 2 on servers: Each hardware type in each server has a total capacity called briefly the Capacity. There is also an in-use value called In Use. For the current job, we filter all servers with the Capacity-In-use value higher than the required value for the job.

$$\Lambda_{\text{HardwareType}}\textbf{VirtualServer} \Rightarrow$$
$$\textbf{(Capacity-Inuse) in VirtualServer} > 0 \quad \Lambda$$
$$\textbf{(Capacity-Inuse) in VirtualServer} > \textbf{(Consumption) in Job} \quad (2)$$

Now it is time to rate servers according Capacity-In use value, which we called remain later on. We will calculate the maximum remain for each hardware type on the servers and,

accordingly, rate each server. That is, the specific score of a hardware type for each hardware type available on servers is calculated as follows:

$$\textbf{Hardware Type Score in VirtualServer} = \frac{\text{(Capacity-Inuse) of Current Hardware Type in Current VirtualServer}}{\max_{\text{Current Hardware Type in All VirtualServers}} \text{(Capacity-Inuse)}} \tag{3}$$

Given the fact that the previous approach leads to the elimination of the unit in computing, we can calculate a server score according to the scores of different hardware types on a server as follows:

$$\textbf{Score of VirtualServer} = \sum_{n}^{\text{Hardware Type in VirtualServer index}=i} \frac{\text{(Capacity-Inuse) of Current Hardware Type in Current VirtualServer}}{\max_{\text{Current Hardware Type in All VirtualServers}} \text{(Capacity-Inuse)}} \tag{4}$$

With regard to the score obtained for each server, we can determine the optimal server for the job. Moreover, if the user selects dynamic variable values for participating in the load balancing process (for each server), the values participated in the computing process according to their data entry style in the software as follows:

$$\textbf{Final Score of VirtualServer} = \sum_{n}^{\text{Hardware Type in VirtualServer index}=i} \text{Hardware Type Score}[i] + \sum_{n}^{\text{Dynamic Variable in VirtualServer index}=i} \text{Dynamic Variable}[i] \tag{5}$$

5. Evaluation Results

5.1. Practical Example

In this part a practical example of the execution of the load balancing process is presented. It is assumed that, according to the settings outlined, the load balancing process is started in the system, using both dynamic and predetermined formulas. The current job specifications (Top job in closed queue) and the available server specifications in server queue are presented in Tab. 1 and Tab. 2, respectively.

Table 1. Job_01 Specifications

Job Name	Hardware Requirements
Job_01	CPU:0.5 Hz RAM: 3 GB HDD: 200 MB

In the first-phase filter, SRV_02 is eliminated due to not having all hardware types required. In the second-phase filter, SRV_01 is eliminated due to having only 1 GB 8-7 RAM hardware type that is less than the current job required RAM hardware type, 3 GB. We now need to calculate the server score between SRV_03 and SRV_04 servers. According to all hardware types that the job requires, we give scores to the servers. Therefore, SRV_03 server for GPU hardware type

Table 2. Specification of Virtual Servers

Server Name	Hardware (Capacity/In use)	Remain	Dynamic Variables (-100<Score>100)
SRV_01	CPU: (3.2 / 2.1) Hz RAM: (8 / 7) GB HDD: (500 / 100) MB	CPU: 1.1 Hz RAM: 1 GB HDD: 400 MB	-
SRV_02	CPU: (3.2 / 0.1) Hz HDD: (5000 / 2500) MB	CPU: 3.1 Hz HDD: 2500 MB	Availability: 90
SRV_03	CPU: (4.1 / 1.2) Hz RAM: (16 / 3) GB HDD: (1000/ 100) MB GPU: (3.1 / 0.1)	CPU: 2.9 Hz RAM: 13 GB HDD: 900 MB GPU: 3	Availability: -63
SRV_04	CPU: (4.1 / 1.8) Hz RAM: (32 / 24) GB HDD: (1500 / 350) MB	CPU: 2.3 Hz RAM: 8 GB HDD: 1150 MB	Communication Delay: 26

is not given any score, because here this hardware type is not needed. The score is calculated between the servers remaining in the queue according to each hardware type based on that hardware type maximum. The score for SRV_03 and SRV_04 is illustrated in Tab. 3.

Table 3. Comparing Score for SRV_03 and SRV_04

Hardware Types In Current Job	Maximum Remain Between All Hardware Type in Servers	Score of SRV_03	Score of SRV_04
CPU	2.9 Hz (SRV_03)	2.9 Hz/2.9 Hz =1	2.3 Hz/2.9 Hz= 0.79
RAM	13 GB (SRV_03)	13 GB/13 GB= 1	8 GB/13 GB= 0.61
HDD	1150 MB (SRV_04)	900 MB/1150 MB =0.78	1150 MB/1150 MB= 1
SUM		**2.78**	**2.39**

So far, the score of each server is calculated based on the proposed algorithm. Now it is time to apply dynamic variable scores to the system. The result is presented in Tab. 4.

As depicted in Fig. 10 it is clear that the winner server is SRV_04.

5.2. Simulation Procedure

In this section, the simulation steps of a load balancing algorithm in the system is presented. We simulate the centralized information algorithm and centralized decision-making algorithm in our proposed dashboard. As mentioned earlier, system information is stored in a single node

Table 4. Calculating SRV_03 and SRV_04 Score in Dynamic Variable Case

Server Name	Dynamic Variable Score	Current Score	Total Score (Dynamic Variable) Current
SRV_03	-63 /100 = -0.63	**2.78**	**2.15**
SRV_04	26 /100 = 0.26	**2.39**	**2.65**

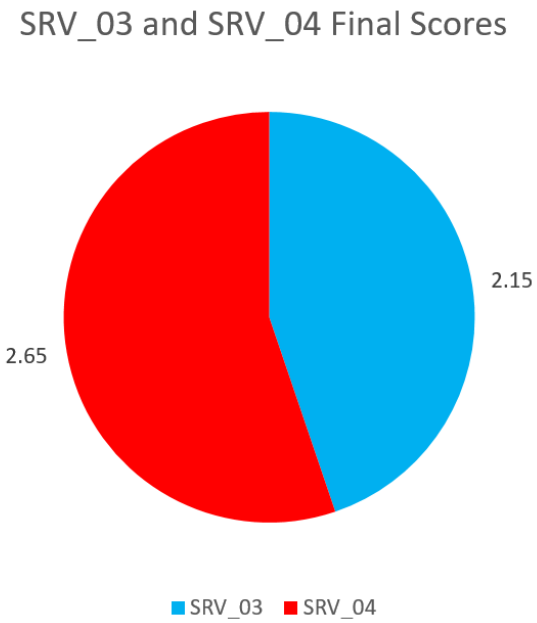


Figure 10. General comparison between SRV_03 and SRV_04

and the decision is applied by the same node in this category of algorithms. The central part is a subset of this algorithm. Then according to the described example, the dashboard can play the role of the center in this system. The developed algorithm is based on load balancing process using dynamic algorithms. At first, we consider some basic rules for the existing environment as follows:

Rule 1: Some services have uncertain latency in the system. However, the latency can be generally divided into three categories based on to the range of changes.

- Communication latency of less than 20 milliseconds;
- Communication latency of 21-80 milliseconds; and
- Communication latency of 81-150 milliseconds.

According to the job type in this environment and the execution time, servers with a latency of more than 151 milliseconds are considered to be rejected, and nothing refers to them.

Rule 2: The custodian of the environment in which the load balancing process is implemented is a subset of a major organ. According to this fact, the organization will always keep its most powerful server in reserve, according by order of the same organ, and this fact must be taken into account for the load balancing process.

Rule 3: Due to financial constraints for the custodian responsible for load balancing, some of these servers are subject to defects. This will make the custodian prefer, in equal conditions, that the job (task) is referred to the servers that are not subject to degradation problems.

Rule 4: Because of the geographical conditions of the area in which the servers are located, they are subject to high humidity. Therefore, humidity control devices are used to control the humidity level for the servers in different racks. Currently, it is preferable not to refer the job to some servers (located on the second floor of the building) which should also be considered in the implementation of the load balancing process. Given the limitations and conditions above, we need to implement this algorithm in the environment in such a way that we cover all the aforementioned aspects.

The Solution for Rule 1: The use of dynamic variables. Given the above and the sensitivity and importance of time, we define a dynamic variable called Communication Delay to apply the corresponding servers by using the following rules:

- Communication Delay is zero for servers with a latency of less than 20 milliseconds. So, we are allowed not to apply this variable to this category of servers because it will not affect the computing results;
- Communication Delay is -20 for servers with a latency of 21-80 milliseconds; and
- Communication Delay is -60 for servers with a latency of 81-150 milliseconds.

The Solution for Rule 2: The use of system settings when developing the resources allocation algorithm. This is anticipated when implementing the resource allocation process in the system. The user determines which server is the winner after the execution of load balancing process in the system. After determining the second server as the winner, the company's most powerful server will be used as reserve.

The Solution for Rule 3: The use of dynamic variables. We can fully cover Rule 3 by defining a dynamic variable called Quality and set it as follows:

- For servers subject to degradation problems, we consider the value of Quality to be -40; and
- For the rest of the servers, we consider the value of this variable to be +40.

The Solution for Rule 4: The use of dynamic variables. We define a dynamic variable called Humidity with a value of -50 for the servers located on the second floor. Now with regard to the above, we can conclude that all the necessary infrastructures required for the implementation of the load balancing process are provided through the proposed solutions.

As depicted in Fig.11, the steps to implement a load balancing algorithm using the proposed system are as follows:

1. Full knowledge of the behavior of the algorithm in terms of both of the load balancing and resource allocation.
2. Implementation of load balancing approach and resource allocation approach using the dashboard facilities.
3. Applying the existing limitations and rules to the system by defining related dynamic variables.

Conclusion

Load balancing is a key necessity to many computing systems. Load balancing not only improves the performance but also decrease the system failure points. In practice, a load balancing algorithm include many different variables. Depending on the type, these variables, can have a wide level of influences on runtime as well as the overall system efficiency. In this research we proposed a flexible dashboard in the environments of load balancing algorithms. In this approach, we improved the flexibility of the system to be adaptive to dynamic variables. We

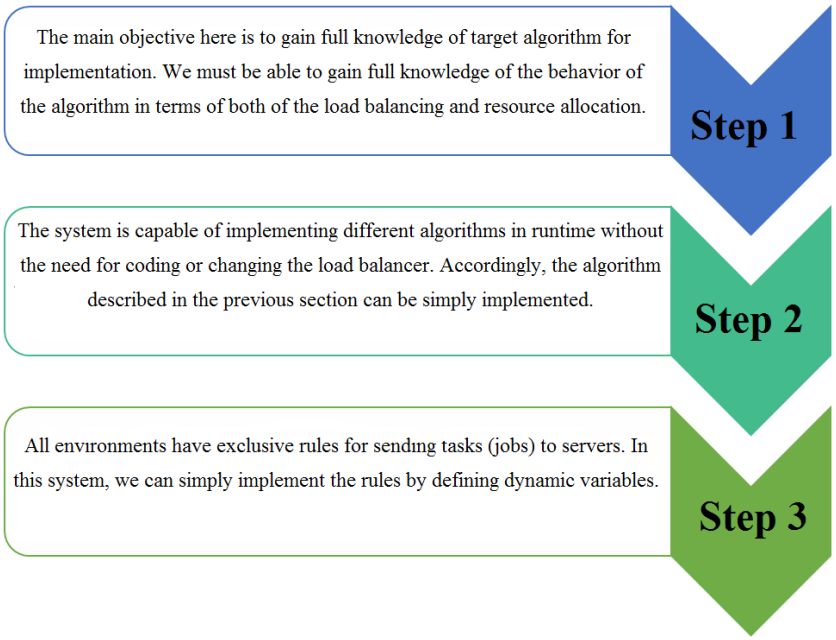


Figure 11. General method for implementing load balancing algorithms in the proposed dashboard

designed a rule-based dashboard to manage the load balancing process in real-time runtime. We proposed a scoring system to find the best computing node based on the task specification in an adaptive manner. We showed that our proposed system is capable if change the boundaries and limitations of the system, changing the load balancing process, introducing new variables and removing the existing ones and measuring the running algorithm in the system. We evaluated our proposed system based at a practical implementation. We also presented a simulation based on different rules in a limited environment. Evaluation results proved that the proposed system reduced the probability of the system failure and also determined the maturity of the system according to the environmental conditions of the load balancing process.

References

1. Arab, M., Mirtaheri, S., Khaneghah, E., Sharifi, M., Mohammadkhani, M.: Improving learning-based request forwarding in resource discovery through load-awareness. Data Management in Grid and Peer-to-Peer Systems pp. 73–82 (2011), DOI: 10.1007/978-3-642-22947-3_7

2. Balasangameshwara, J., Raju, N.: Performance-driven load balancing with a primary-backup approach for computational grids with low communication cost and replication cost. IEEE Transactions on Computers 62(5), 990–1003 (2013), DOI: 10.1109/TC.2012.44

3. Brodowicz, M., Sterling, T.: Simultac fonton: A fine-grain architecture for extreme performance beyond moore's law. Supercomputing Frontiers and Innovations 4(2), 27–37 (2017), DOI: 10.1109/HPCSim.2016.7568352

4. Casanova, H.: Simgrid: A toolkit for the simulation of application scheduling. In: Cluster computing and the grid, 2001. Proceedings. First IEEE/ACM international symposium on. IEEE (2001), DOI: 10.1109/CCGRID.2001.923223

5. Dinitz, M., Fineman, J., Gilbert, S., Newport, C.: Load balancing with bounded convergence in dynamic networks. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. pp. 1–9 (2017), DOI: 10.1109/INFOCOM.2017.8057000

6. Domanal, S.G., Reddy, G.R.M.: Load balancing in cloud environment using a novel hybrid scheduling algorithm. In: Cloud Computing in Emerging Markets (CCEM), 2015 IEEE International Conference on. pp. 37–42. IEEE (2015), DOI: 10.1109/CCEM.2015.31

7. Heene, M., Kowitz, C., Pflüger, D.: Load balancing for massively parallel computations with the sparse grid combination technique. In: PARCO. pp. 574–583 (2013), DOI: 10.3233/978-1-61499-381-0-574

8. LD, D.B., Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing 13(5), 2292–2303 (2013), DOI: 10.1016/j.asoc.2013.01.025

9. Mahafzah, B.A., Jaradat, B.A.: The hybrid dynamic parallel scheduling algorithm for load balancing on chained-cubic tree interconnection networks. The Journal of Supercomputing 52(3), 224–252 (2010), DOI: 10.1007/s11227-009-0288-3

10. Mirtaheri, S.L., Grandinetti, L.: Dynamic load balancing in distributed exascale computing systems. Cluster Computing (May 2017), DOI: 10.1007/s10586-017-0902-8

11. Mohamed, N., Al-Jaroodi, J.: Delay-tolerant dynamic load balancing. In: 2011 IEEE International Conference on High Performance Computing and Communications. pp. 237–245 (2011), DOI: 10.1109/HPCC.2011.39

12. Mousavi Khaneghah, E., Mirtaheri, S.L., Sharifi, M., Minaei Bidgoli, B.: Modeling and analysis of access transparency and scalability in p2p distributed systems. International Journal of Communication Systems 27(10), 2190–2214 (2014), DOI: 10.1002/dac.2467

13. Patel, P., Bansal, D., Yuan, L., Murthy, A., Greenberg, A., Maltz, D.A., Kern, R., Kumar, H., Zikos, M., Wu, H., et al.: Ananta: Cloud scale load balancing. In: ACM SIGCOMM Computer Communication Review. vol. 43, pp. 207–218. ACM (2013), DOI: 10.1145/2486001.2486026

14. Pickartz, S., Lankes, S., Monti, A., Clauss, C., Breitbart, J.: Application migration in HPC driver of the exascale era? In: High Performance Computing & Simulation (HPCS), 2016 International Conference on. pp. 318–325. IEEE (2016), DOI: 10.1109/HPCSim.2016.7568352

15. Sharma, M., Yadav, A., Sharma, P.: An optimistic approach for load balancing in cloud computing pp. 27–30 (2014), international Journal of Computer Science and Engineering 2(3)

16. Soltani, N., Khaneghah, E.M., Sharifi, M., Mirtaheri, S.L.: A dynamic popularity-aware load balancing algorithm for structured p2p systems. Springer (2012), DOI: 10.1007/978-3-642-35606-3_9

17. Wang, K., Zhou, X., Qiao, K., Lang, M., McClelland, B., Raicu, I.: Towards scalable distributed workload manager with monitoring-based weakly consistent resource stealing. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing. pp. 219–222. ACM (2015), DOI: 10.1145/2749246.2749249

18. Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., Raicu, I.: Optimizing load balancing and data-locality with data-aware scheduling. In: Big Data (Big Data), 2014 IEEE International Conference on. pp. 119–128. IEEE (2014), DOI: 10.1109/BigData.2014.7004220

19. Xu, G., Pang, J., Fu, X.: A load balancing model based on cloud partitioning for the public cloud. Tsinghua Science and Technology 18(1), 34–39 (2013), DOI: 10.1109/TST.2013.6449405

DOI: 10.14529/jsfi170404

Additivity: A Selection Criterion for Performance Events for Reliable Energy Predictive Modeling

Arsalan Shahid[1] **, *Muhammad Fahad***[1]**, *Ravi Reddy***[1]**, *Alexey Lastovetsky***[1]

Performance events or performance monitoring counters (PMCs) are now the dominant predictor variables for modeling energy consumption. Modern hardware processors provide a large set of PMCs. Determination of the best subset of PMCs for energy predictive modeling is a non-trivial task given the fact that all the PMCs can not be determined using a single application run. Several techniques have been devised to address this challenge. While some techniques are based on a statistical methodology, some use expert advice to pick a subset (that may not necessarily be obtained in one application run) that, in experts' opinion, are significant contributors to energy consumption. However, the existing techniques have not considered a fundamental property of predictor variables that should have been applied in the first place to remove PMCs unfit for modeling energy. We address this oversight in this paper.

We propose a novel selection criterion for PMCs called *additivity*, which can be used to determine the subset of PMCs that can potentially be used for reliable energy predictive modeling. It is based on the experimental observation that the energy consumption of a serial execution of two applications is the sum of energy consumptions observed for the individual execution of each application. A linear predictive energy model is consistent if and only if its predictor variables are additive in the sense that the vector of predictor variables for a serial execution of two applications is the sum of vectors for the individual execution of each application. The criterion, therefore, is based on a simple and intuitive rule that the value of a PMC for a serial execution of two applications is equal to the sum of its values obtained for the individual execution of each application. The PMC is branded as *non-additive* on a platform if there exists an application for which the calculated value differs significantly from the value observed for the application execution on the platform. The use of *non-additive* PMCs in a model renders it inconsistent.

We study the *additivity* of PMCs offered by the popular state-of-the-art tools, *Likwid* and *PAPI*, by employing a detailed experimental methodology on a modern Intel Haswell multicore server CPU. We show that many PMCs in *Likwid* and *PAPI* that are widely used in models as key predictor variables are *non-additive*. This brings into question the reliability and the reported prediction accuracy of these models.

Keywords: performance events, PMC, energy predictive models, Likwid, PAPI.

Introduction

Performance events or performance monitoring counters (PMCs) are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. We will use the acronym PMCs to refer to software events, which are pure kernel-level counters such as *page-faults*, *context-switches*, etc. as well as micro-architectural events originating from the processor and its performance monitoring unit called the hardware events such as *cache-misses*, *branch-instructions*, etc. They have been developed primarily to aid low-level performance analysis and tuning. Remarkably while PMCs have not been used for performance modeling, over the years, they have become dominant predictor variables for energy predictive modeling.

Modern hardware processors provide a large set of PMCs. Consider the Intel Haswell multicore server CPU whose specification is shown in Tab. 1. On this server, the PAPI tool [19] provides 53 hardware performance events. The Likwid tool [17, 23] provides 167 PMCs. This

[1]School of Computer Science, University College Dublin, Dublin, Ireland

Table 1. Specification of the Intel Haswell Multicore CPU

Technical Specifications	**Intel Haswell Server**
Processor	Intel E5-2670 v3 @2.30GHz
OS	CentOS 7
Micro-architecture	Haswell
Thread(s) per core	2
Cores per socket	12
Socket(s)	2
NUMA node(s)	2
L1d cache	32 KB
L11 cache	32 KB
L2 cache	256 KB
L3 cache	30720 KB
Main memory	64 GB DDR4
Memory bandwidth	68 GB/sec
TDP	240 W
Idle Power	58 W

includes events for uncore and micro-operations (μops) of CPU cores specific to Haswell architecture that are not provided by PAPI. However, all the PMCs can not be determined using a single application run since only a limited number of registers is dedicated to collecting them. For example, to collect all the Likwid PMCs for a single runtime configuration of an application on the server, the application must be executed 53 times. It must be also pointed out that energy predictive models based on PMCs are not portable across a wide range of architectures. While a model based on either Likwid PMCs or PAPI PMCs may be portable across Intel and AMD architectures, it will be unsuitable for GPU architectures.

Therefore, there are three serious constraints that pose difficult challenges to employing PMCs as predictor variables for energy predictive modeling. First, there is a large number of PMCs to consider. Second, tremendous programming effort and time are required to automate and collect all the PMCs. This is because all the PMCs can not be collected in one single application run. Third, a model purely based on PMCs lacks portability. In this paper, we focus mainly on techniques employed to select a subset of PMCs to be used as predictor variables for energy predictive modeling. We now present a brief survey of them.

O'Brien et al. [18] survey the state-of-the-art energy predictive models in HPC and present a case study demonstrating the ineffectiveness of the dominant PMC-based modeling approach for accurate energy predictions. In the case study, they use 35 carefully selected PMCs (out of a total of 390 available in the platform) in their linear regression model for predicting dynamic energy consumption. [1, 9, 10] select PMCs manually, based on in-depth study of architecture and empirical analysis. [8, 15, 18, 21, 22, 27, 30] select PMCs that are highly correlated with energy consumption using Spearman's rank correlation coefficient (or Pearson's correlation coefficient) and principal component analysis (PCA). [1, 2, 15] use variants of linear regression to remove PMCs that do not improve the average model prediction error.

From the survey, we can classify the existing techniques into three categories. The first category contains techniques that consider all the PMCs with the goal to capture all possible

contributors to energy consumption. To the best of our knowledge, we found no research works that adopt this approach. This could be due to several reasons: a) Gathering all PMCs requires huge programming effort and time; b) Interpretation (for example, visual) of the relationship between energy consumption and PMCs is difficult especially when there is a large number of PMCs; c) Dynamic or runtime models must choose PMCs that can be gathered in just one application run; d) Typically, simple models (those with less parameters) are preferred over complex models not because they are accurate but because simplicity is considered a desirable virtue.

The second category consists of techniques that are based on a statistical methodology. The last category contains techniques that use expert advice or intuition to pick a subset (that may not necessarily be determined in one application run) and that, in experts' opinion, is a dominant contributor to energy consumption. However, the existing techniques have not considered one fundamental property of predictor variables that should have been considered in the first place to remove PMCs unfit for modeling energy. We address this oversight in this paper.

We propose a novel selection criterion for PMCs called *additivity*, which can be used to determine the subset of PMCs that can potentially be used for reliable energy predictive modeling. It is based on the experimental observation that the energy consumption of a serial execution of two applications is the sum of energy consumptions observed for the individual execution of each application. We define a *compound application* to represent a serial execution of a combination of two or more individual applications. The individual applications are also termed as *base applications*.

A linear predictive energy model is consistent if and only if its predictor variables are additive in the sense that the vector of predictor variables for a compound application is the sum of vectors for the individual execution of each application. The *additivity* criterion, therefore, is based on simple and intuitive rule that the value of a PMC for a compound application is equal to the sum of its values for the executions of the base applications constituting the compound application.

We brand a PMC *non-additive* on a platform if there exists a compound application for which the calculated value significantly differs from the value observed for the application execution on the platform (within a tolerance of 5.0%). If we fail to find a compound application (typically from a sufficiently large suite of compound applications) for which the *additivity* criterion is not satisfied, we term the PMC as potentially *additive*, which means that it can potentially be used for reliable energy predictive modeling. By definition, a potentially *additive* PMC must be deterministic and reproducible, that is, it must exhibit the same value (within a tolerance of 5.0%) for different executions of the same application with same runtime configuration on the same platform.

The use of a *non-additive* PMC as a predictor variable in a model renders it inconsistent and therefore unreliable.

We study the *additivity* of PMCs offered by two popular tools, *Likwid* and *PAPI*, by employing a detailed statistical experimental methodology on a modern Intel Haswell multicore server CPU. We observe that all the Likwid PMCs and PAPI PMCs are reproducible. However, we show that while many PMCs are potentially *additive*, a considerable number of PMCs are not. Some of the *non-additive* PMCs are widely used in energy predictive models as key predictor variables.

For each *non-additive* PMC, we determine the maximum percentage error (averaged over several runs) observed experimentally. This is the ratio of the difference between the PMC of

a compound application and the sum of the PMCs of the base applications and the sum of the PMCs. We show that there is a PMC where the error is as high as 3075% and there are several PMCs where the error is over 100%. This brings into question the reliability and reported prediction accuracy of models that use these PMCs.

Our key contribution in this work is that we propose a novel criterion called *additivity* that can be used to identify PMCs not suitable for energy predictive modeling. PMCs offered by popular tools such as *Likwid* and *PAPI* are classified based on this criterion using a detailed experimental methodology on a modern Intel Haswell multicore server CPU. In our future work, we plan to classify the *non-additivity* of PMC into application-specific and platform-specific categories.

The rest of the paper is structured as follows. Section 1 surveys popular tools that provide programmatic and command-line interfaces to obtain PMCs. In Section 2, we define the property of *additivity* and explain why it is important for reliable energy predictive modeling. Section 3 presents the experimental methodology used to determine the *additivity* of Likwid and PAPI PMCs. Sections 4 and 5 present a classification of Likwid and PAPI PMCs respectively based on the criterion of *additivity*. Finally, Section 6 concludes the paper.

1. Related Work

This section is divided into two parts. In the first part, we present tools widely used to obtain PMCs. In the second part, we survey notable research on selection of PMCs for power and energy modeling from a large set supplied by a tool.

1.1. Tools to Determine PMCs

PAPI [19] provides a standard API for accessing PMCs available on most modern microprocessors. It provides two types of events, *native events* and *preset events*. *Native events* correspond to PMCs native to a platform. They form the building blocks for *preset events*. A *preset event* is mapped onto one or more native events on each hardware platform. While *native events* are specific to each platform, *preset events* obtained on different platforms can not be compared.

Likwid [22] provides command-line tools and an API to obtain PMCs for both Intel and AMD processors on the Linux OS.

For Nvidia GPUs, CUDA Profiling Tools Interface (*CUPTI*) [3] can be used for obtaining the PMCs. *Intel PCM* [14] is used for reading PMCs of core and uncore (which includes the QPI) components of an Intel processor. *Perf* [25] also called *perf_events* can be used to gather the PMCs for CPUs in Linux.

1.2. Techniques for Selection of PMCs for Energy Predictive Modeling

All the models surveyed in this section are linear energy predictive models.

Singh et al. [20] use PMCs provided by AMD Phenom processor. They divide the PMCs into four categories and rank them in the increasing order of correlation with power using the Spearman's rank correlation. Then they select the top PMC in each category (four in total) for their energy prediction model.

Goel et al. [8] divide PMCs into event categories that they believe capture different kinds of microarchitectural activity. The PMCs in each category are then ordered based on their

correlation to power consumption using the Spearman's rank correlation. The PMCs with less correlation are then investigated by analyzing the accuracy of several models that employ them.

Kadayif et al. [16] present a PMC-based model for predicting energy consumption of programs on a UltraSPARC platform. The platform provides 30 different PMCs. However, they use only eight and do not specify how they have selected them.

Lively et al. [17] employ 40 PMCs in their predictive model. They use an elaborate statistical methodology to select PMCs. They compute the Spearman's rank correlation for each PMC and remove those below a threshold. They compute the principal components (PCA) of the remaining PMCs and select those with the highest PCA coefficients. Bircher et al. [1] employ an iterative linear regression modeling process where they add a PMC at each step and stop until desired average prediction error is achieved.

Song et al. [21] select a group of PMCs (for their energy model of Nvidia Fermi C2075 GPU) that are strongly correlated to power consumption based on the Pearson correlation coefficient.

Witkowski et al. [26] use PMCs provided by the *Perf* tool for their model. They use the correlation (Pearson correlation coefficient) between a PMC and the measured power consumption and select those PMCs, which have high correlation coefficients. Although they find that the PMCs related to DRAM have a low correlation with power consumption, they still use them since these variables signify intensity of DRAM operations, which contribute significantly to power consumption.

Gschwandtner et al. [9] deal with the problem of selecting the best subset of PMCs on the IBM POWER7 processor, which offers over 500 different PMCs. They first manually select a medium number of hardware counters that they believe are prominent contributors to energy consumption. Then they empirically select a subset from their initial selection. Jarus et al. [15] use PMCs provided by the *Perf* tool for their models. The PMCs employed differ for different models and are selected using two-stage process. In the first stage, PMCs that are correlated 90% or above are selected. In the second stage, stepwise regression with forward selection is used to decide the final set of PMCs.

Haj-Yihia et al. [10] start with a set of 23 PMCs (offered by Likwid) based on expert knowledge of the Intel architecture. Then they perform linear regression iteratively where they drop PMCs (one by one) that do not impact the average prediction error of their model.

Wu et al. [29] use the Spearman correlation coefficient and PCA to select the subset of PMCs, that are highly correlated with power consumption. Chadha et al. [2] select a particular PMC from the list of PAPI PMCs available for their platform and check if it fits well with linear regression model. If it does, they select it as a key parameter for their modeling and experimental study. Otherwise, they skip it.

2. Additivity: Definition

The *additivity* criterion is based on simple and intuitive rule that the value of a PMC for a compound application is equal to the sum of its values for the executions of the base applications constituting the compound application.

We brand a PMC *non-additive* on a platform if there exists a compound application for which the calculated value significantly differs from the value observed for the application execution on the platform (within a tolerance of 5.0%). If the experimentally observed PMCs (sample means) of two base applications are $\overline{e_1}$ and $\overline{e_2}$ respectively, then a *non-additive* PMC of the compound

Table 2. List of Applications

Application	Description
NPB IS	Integer Sort, Kernel for random memory access
NPB LU	Lower-Upper Gauss-Seidel solver
NPB EP	Embarrassingly Parallel, Kernel
NPB BT	Block Tri-diagonal solver
NPB MG	Multi-Grid on a sequence of meshes
NPB FT	Discrete 3D fast Fourier Transform
NPB DC	Data Cube
NPB UA	Unstructured Adaptive mesh, dynamic and irregular memory access
NPB CG	Conjugate Gradient
NPB SP	Scalar Penta-diagonal solver
NPB DT	Data traffic
MKL FFT	Fast Fourier Transform
MKL DGEMM	Dense Matrix Multiplication
HPCG	High performance conjugate gradient
stress	CPU, disk and I/O stress
Naive MM	Naive Matrix-matrix multiplication
Naive MV	Naive Matrix-vector multiplication

application will experimentally exhibit a count that does not lie between $(\overline{e_1} + \overline{e_2}) \times (1 - \epsilon)$ and $(\overline{e_1} + \overline{e_2}) \times (1 + \epsilon)$, where the tolerance $\epsilon = 0.05$.

If we fail to find a compound application (typically from a large set of diverse compound applications) for which the *additivity* criterion fails, we term the PMC as potentially *additive*, which means that it can potentially be used for reliable energy predictive modeling. By definition, a potentially *additive* PMC must be deterministic and reproducible, that is, it must exhibit the same value (within a tolerance of 5.0%) for different executions of the same application with the same runtime configuration on the same platform.

The use of a *non-additive* PMC as a predictor variable in a model renders it inconsistent and therefore unreliable. We explain this point using a simple example. Consider an instance of an energy prediction model that uses a *non-additive* PMC as a predictor variable. A natural and intuitive approach to predict the energy consumption of an application that executes two base applications one after the other is to substitute the sum of the PMCs for the base applications in the model. However, since the PMC is *non-additive*, the prediction would be very inaccurate.

Therefore, using *non-additive* PMCs in energy predictive models adds noise and can significantly damage the predicting power of energy models based on them.

We now present a test to determine if a PMC is *non-additive* or potentially *additive*. We call it the *additivity test*.

2.1. Additivity Test

The test consists of two stages. A PMC must pass both stages to be declared *additive* for a given compound application on a given platform. At the first stage, we determine if the PMC is deterministic and reproducible.

At the second stage, we examine how the PMC of the compound application relates to its values for the base applications. At first, we collect the values of the PMC for the base applications by executing them separately. Then, we execute the *compound* application and obtain its value of the PMC. Typically, the core computations for the compound application consist of the core computations of the base applications programmatically placed one after the other. This has to be the case for PAPI PMCs. However, for Likwid PMCs, one can use the *system* call to invoke the base application. It must also be ensured that the execution of the *compound* application takes place under platform conditions similar to those for the execution of its constituent *base* applications.

If the PMC of the *compound* application is equal to the sum of the PMCs of the base applications (with a tolerance of 5.0%), we classify the PMC as potentially *additive*. Otherwise, it is *non-additive*.

We call the PMC that passes the *additivity test* potentially additive. For it to be called *absolutely additive* on a platform, ideally it must pass the test for all conceivable compound applications on the platform. Therefore, we avoid this definition.

In our experiments, we observed that all the PMCs were deterministic and reproducible.

For each PMC, we determine the maximum percentage error. For a *compound* application, the percentage error (averaged over several runs) is calculated as follows:

$$Error(\%) = (|\frac{(\overline{e_{b1}} + \overline{e_{b2}}) - \overline{e_c}}{\overline{e_{b1}} + \overline{e_{b2}}}|) \times 100 \quad (1)$$

where $\overline{e_c}, \overline{e_{b1}}, \overline{e_{b2}}$ are the sample means of predictor variables for the compound application and the constituent base applications respectively. The maximum percentage error is then calculated as the maximum of the errors for all the *compound* applications in the experimental testsuite.

3. Experimental Methodology to Obtain Likwid and PAPI PMCs

In this section, we present our experimental setup to determine the *additivity* of PMCs.

The experiments are performed on the Intel Haswell multicore CPU platform (specifications given in Tab. 1). We used diverse range of applications (both compute-bound and memory-bound) in our testsuite composed of NAS parallel benchmarking suite (NPB), Intel math kernel library (MKL), HPCG [13], and *stress* [24] (description given in Tab. 2). The experimental workflow is shown in Fig. 1 where the internals of the server are shown in great detail.

For each run of a application in our testsuite, we measure the following: 1) Dynamic energy consumption, 2) Execution time, and 3) PMCs. The dynamic energy consumption and the application execution time are obtained using the HCLWattsUp interface [11]. We would like to mention that the output variables (or response variables) in the performance and energy predictive models, i.e. energy consumption and execution time, are *additive*. We confirm this via thorough experimentation and therefore we will not discuss them hereafter.

We now present our experimental methodologies for determining Likwid and PAPI PMCs.

3.1. LIKWID PMCs

In this section, we explain the experimental methodology to obtain Likwid PMCs.

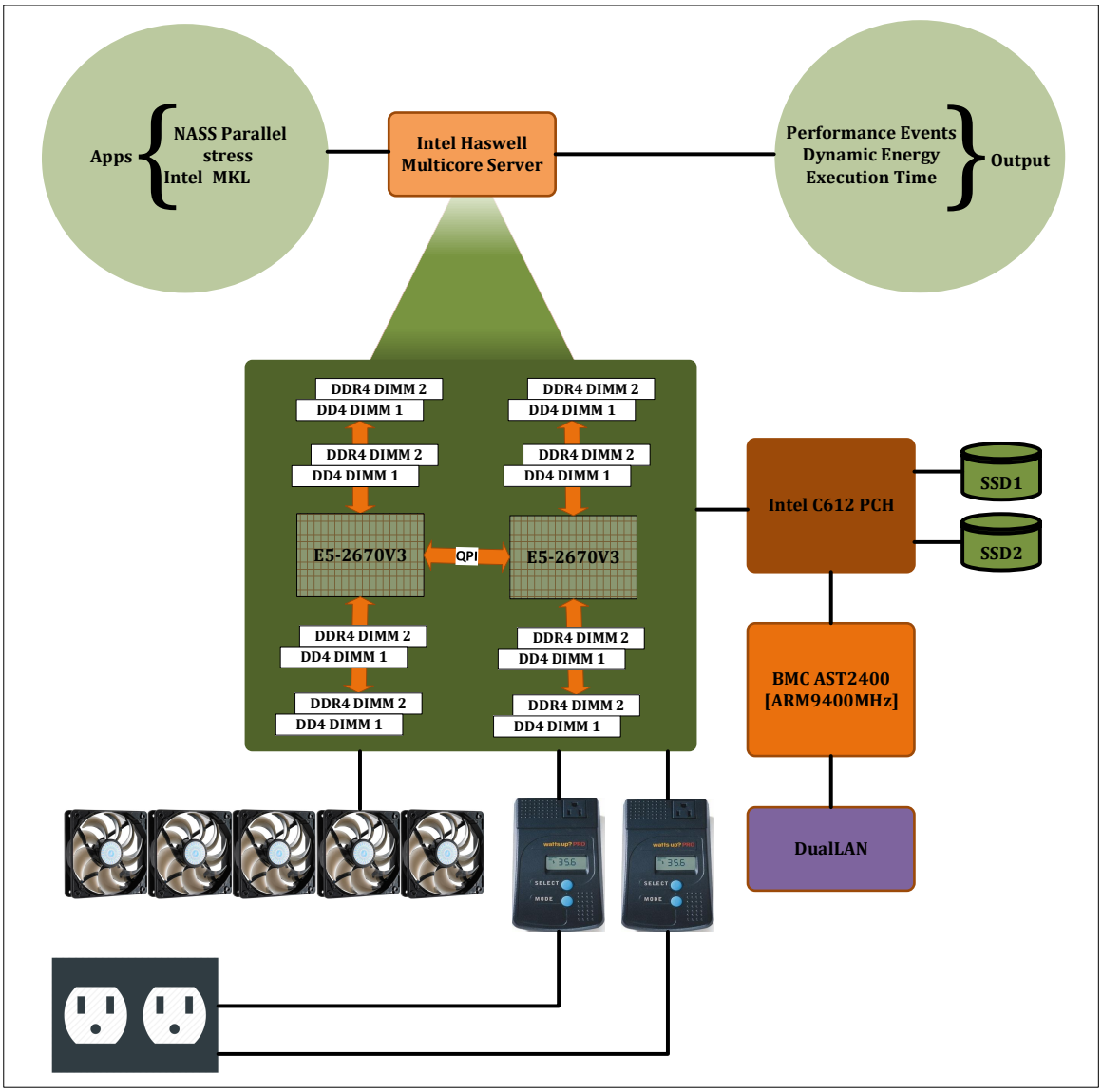


Figure 1. Experimental workflow to determine the PMCs on the Intel Haswell server.

A sample Likwid command-line invocation is shown below where *EVENTS* represents one or more PMCs, which are collected during the execution of the given application *APP*:

likwid-perfctr -f -C S0:0-11@S1:12-23 -g EVENTS APP

Here, the application (*APP*) during its execution is pinned to physical cores (0-11, 12-23) in our platform. Since Likwid does not provide option to bind application to memory, we have used *numactl*, i.e. a command-line linux tool, with option *–membind* to pin our applications to memory blocks (for our platform **numactl** gives 2 memory blocks, 0 and 1). The list of comma-separated PMCs is specified in *EVENTS*. For example, the following command:

likwid-perfctr -f -C S0:0-11@S1:12-23
-g ICACHE_ACCESSES:PMC0,ICACHE_MISSES:PMC1
numactl –membind=0,1 APP

determines the counts for two PMCs, *ICACHE_ACCESSES:PMC0* and *ICACHE MISSES:PMC1*.

Collection of all PMCs requires significant programming efforts and execution time because only a limited number of PMCs can be obtained in a single application run due to the limited

number of registers dedicated to collecting PMCs. In addition, to ensure the reliability of our results, we follow a detailed statistical methodology where sample mean of a PMC is used. It is calculated by executing the application repeatedly until it lies in the 95% confidence interval and a precision of 0.050 (5.0%) has been achieved. For this purpose, Student's t-test is used assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions by plotting the distributions of observations.

Likwid provides 167 PMCs for our platform. In order to collect all of them for an application, we have to run the application 53 times. We wrote a software tool to automate this collection process, *SLOPE-PMC-LIKWID* [12].

Before we apply the *additivity* test, we remove few PMCs such as IIO_CREDIT (related to I/O and QPI), and $OFFCORE\_RESPONSE$ since they exhibit zero counts. We also remove PMCs having very low count (less than 10). The resulting dataset contained 151 performance events, which are then input to the *additivity* test.

3.2. PAPI PMCs

In this section, we explain the experimental methodology to obtain PAPI PMCs.

We check the available PAPI PMCs for our Intel Haswell platform using the command-line invocation, $'papi\_avail - a'$. We found that a total of 53 PMCs are available. The number of PMCs that can be gathered in a single application run varies. While gathering a set of 4 PMCs is common, there are a few event sets, which can contain up to 2 or 3 PMCs. Therefore, we found that the application has to be executed 14 times in order to collect all the PMCs for the application on our platform.

We wrote a software tool to automate the process of collection of PMCs, SLOPE-PMC-PAPI [12]. It is to be noted that for ensuring the reliability of our experimental results, we follow the same statistical methodology that was followed for determining Likwid PMCs.

4. Additivity of Likwid PMCs

In this section, we determine the *additivity* of Likwid PMCs. We execute all the *compound* applications where each application is composed of two base applications in our testsuite (shown in Tab. 2).

The list of potentially *additive* PMCs is shown in the Tab. 3. The list of *non-additive* PMCs is presented in Tab. 4, which also reports the maximum percentage error for each PMC.

It is noteworthy that some *non-additive* PMCs are used as predictor variables in many energy predictive models [5, 6, 10, 20, 23]. These are *ICache events*, *L2 Transactions*, and *L2 Requests*.

5. Additivity of PAPI PMCs

In this section, we determine the *additivity* of PAPI PMCs. We again execute all the *compound* applications where each application is composed of two base applications in our testsuite (shown in Tab. 2).

The list of potentially *additive* PMCs is shown in Tab. 5. The list of *non-additive* PMCs is shown in Tab. 6, which also reports the maximum percentage error for each PMC.

Table 3. List of Potentially *Additive* Likwid PMCs

BR_INST_EXEC_ALL_BRANCHES	IDQ_UOPS_NOT_DELIVERED_CYCLES_0 _UOPS_DELIV_CORE
BR_MISP_EXEC_ALL_BRANCHES	IDQ_UOPS_NOT_DELIVERED_CYCLES_FE_WAS_OK
BR_INST_RETIRED_ALL_BRANCHES	UOPS_EXECUTED_PORT_PORT_0
BR_MISP_RETIRED_ALL_BRANCHES	UOPS_EXECUTED_PORT_PORT_1
DRAM_CLOCKTICKS	UOPS_EXECUTED_PORT_PORT_2
SNOOPS_RSP_AFTER_DATA_LOCAL	UOPS_EXECUTED_PORT_PORT_3
SNOOPS_RSP_AFTER_DATA_REMOTE	UOPS_EXECUTED_PORT_PORT_4
RXL_FLITS_G1_DRS_NONDATA	UOPS_EXECUTED_PORT_PORT_5
RXL_FLITS_G0_NON_DATA	UOPS_EXECUTED_PORT_PORT_6
TXL_FLITS_G0_NON_DATA	UOPS_EXECUTED_PORT_PORT_7
CPU_CLK_UNHALTED_ANY	UOPS_EXECUTED_PORT_PORT_0_CORE
CPU_CLOCK_UNHALTED_THREAD_P	UOPS_EXECUTED_PORT_PORT_1_CORE
CPU_CLOCK_UNHALTED_THREAD_P_ANY	UOPS_EXECUTED_PORT_PORT_2_CORE
CPU_CLOCK_UNHALTED_REF_XCLK	UOPS_EXECUTED_PORT_DATA_PORTS
CPU_CLOCK_UNHALTED_REF_XCLK_ANY	L2_RQSTS_ALL_DEMAND_REFERENCES
HA_R2_BL_CREDITS_EMPTY_LO_HA0	L2_RQSTS_L2_PF_MISS
HA_R2_BL_CREDITS_EMPTY_LO_HA1	MEM_UOPS_RETIRED_ALL
CPU_CLOCK_THREAD_UNHALTED _ONE_THREAD_ACTIVE	UOPS_EXECUTED_PORT_PORT_3_CORE
CPU_CLOCK_UNHALTED_TOTAL_CYCLES	UOPS_EXECUTED_PORT_PORT_4_CORE
OFFCORE_REQUESTS_OUTSTANDING _DEMAND_DATA_RD	UOPS_EXECUTED_PORT_PORT_5_CORE
OFFCORE_REQUESTS_OUTSTANDING _CYCLES_WITH_DATA_RD	UOPS_EXECUTED_PORT_PORT_6_CORE
OFFCORE_REQUESTS_OUTSTANDING _DEMAND_DATA_RD_C6	UOPS_EXECUTED_PORT_PORT_7_CORE
UOPS_EXECUTED_PORT_DATA_PORTS	UOPS_EXECUTED_PORT_ARITH_PORTS
OFFCORE_REQUESTS_DEMAND_DATA_RD	UOPS_EXECUTED_PORT_ARITH_PORTS_CORE
HA_R2_BL_CREDITS_EMPTY_HI_R2_NCB	UOPS_EXECUTED_PORT_DATA_PORTS
CPU_CLOCK_UNHALTED_THREAD_P	UOPS_RETIRED_CORE_TOTAL_CYCLES
CPU_CLOCK_UNHALTED_THREAD_P_ANY	LSD_CYCLES_4_UOPS
CPU_CLOCK_UNHALTED_REF_XCLK	UOPS_EXECUTED_THREAD
CPU_CLOCK_UNHALTED_REF_XCLK_ANY	UOPS_EXECUTED_USED_CYCLES
CPU_CLOCK_THREAD_UNHALTED _ONE_THREAD_ACTIVE	UOPS_EXECUTED_STALL_CYCLES
CPU_CLOCK_UNHALTED_TOTAL_CYCLES	UOPS_EXECUTED_TOTAL_CYCLES
ICACHE_MISSES	UOPS_EXECUTED_CYCLES_GE_1_UOPS_EXEC
L2_RQSTS_RFO_MISS	UOPS_EXECUTED_CYCLES_GE_2_UOPS_EXEC
L2_RQSTS_ALL_RFO	UOPS_EXECUTED_CYCLES_GE_3_UOPS_EXEC
L2_RQSTS_CODE_RD_HIT	UOPS_EXECUTED_CYCLES_GE_4_UOPS_EXEC
L2_RQSTS_CODE_RD_MISS	UOPS_EXECUTED_CORE
UOPS_EXECUTED_PORT_DATA_PORTS	UOPS_EXECUTED_CORE_USED_CYCLES
MEM_LOAD_UOPS_RETIRED_ALL_ALL	UOPS_EXECUTED_CORE_STALL_CYCLES
UOPS_ISSUED_ANY	UOPS_EXECUTED_CORE_TOTAL_CYCLES
UOPS_ISSUED_USED_CYCLES	UOPS_EXECUTED_CORE_CYCLES_GE_1_UOPS_EXEC
UOPS_ISSUED_STALL_CYCLES	UOPS_EXECUTED_CORE_CYCLES_GE_2_UOPS_EXEC
UOPS_ISSUED_TOTAL_CYCLES	UOPS_EXECUTED_CORE_CYCLES_GE_3_UOPS_EXEC
UOPS_ISSUED_CORE_USED_CYCLES	UOPS_EXECUTED_CORE_CYCLES_GE_4_UOPS_EXEC
UOPS_ISSUED_CORE_STALL_CYCLES	UOPS_RETIRED_ALL
UOPS_ISSUED_CORE_TOTAL_CYCLES	UOPS_RETIRED_CORE_ALL
IDQ_MITE_ALL_UOPS	UOPS_RETIRED_RETIRE_SLOTS
IDQ_DSB_UOPS	UOPS_RETIRED_CORE_RETIRE_SLOTS
IDQ_MS_UOPS	UOPS_RETIRED_USED_CYCLES
IDQ_ALL_DSB_CYCLES_ANY_UOPS	UOPS_RETIRED_STALL_CYCLES
IDQ_ALL_DSB_CYCLES_4_UOPS	UOPS_RETIRED_TOTAL_CYCLES
IDQ_ALL_MITE_CYCLES_ANY_UOPS	UOPS_RETIRED_CORE_USED_CYCLES
IDQ_UOPS_NOT_DELIVERED_CORE	UOPS_RETIRED_CORE_STALL_CYCLES
CAS_COUNT_RD	CAS_COUNT_WR
CAS_COUNT_ALL	

It should be mentioned that some of these *non-additive* PMCs such as $PAPI\_L1\_ICM$ and $PAPI\_L2\_ICM$ have been widely used in energy and performance predictive models [2, 4, 7, 17, 27, 28]. These represent L1 and L2 instruction cache misses.

Table 4. List of *Non-additive* Likwid PMCs

Event Name	Maximum Percentage Error (%)
UNCORE_CLOCK	16.98
CBOX_CLOCKTICKS	16.98
SBOX_CLOCKTICKS	17.08
WBOX_CLOCKTICKS	17.57
BBOX_CLOCKTICKS	16.98
PBOX_CLOCKTICKS	16.98
RBOX_CLOCKTICKS	16.98
QBOX_CLOCKTICKS	17.57
HA_R2_BL_CREDITS_EMPTY_LO_R2_NCB	45.27
HA_R2_BL_CREDITS_EMPTY_LO_R2_NCS	48.28
HA_R2_BL_CREDITS_EMPTY_HI_HA0	203.15
HA_R2_BL_CREDITS_EMPTY_HI_HA1	213.15
HA_R2_BL_CREDITS_EMPTY_HI_R2_NCS	250.56
OFFCORE_RESPONSE_0_DMND_DATA_RD_ANY	47.50
ICACHE_IFETCH_STALL	86.60
L2_RQSTS_RFO_HIT	27.44
ARITH_DIVIDER_UOPS	3075.23
IDQ_UOPS_NOT_DELIVERED_CYCLES_LE_1_UOP_DELIV_CORE	163.64
IDQ_UOPS_NOT_DELIVERED_CYCLES_LE_2_UOP_DELIV_CORE	89.16
L2_RQSTS_L2_PF_HIT	39.41
ICACHE_HIT	105.45
RXL_FLITS_G0_DATA	176.62
OFFCORE_REQUESTS_OUTSTANDING_ALL_DATA_RD	33.76
OFFCORE_REQUESTS_ALL_DATA_RD	42.45
IDQ_MITE_UOPS	42.06
L2_RQSTS_ALL_DEMAND_DATA_RD	52.76
L2_TRANS_DEMAND_DATA_RD	24.29
L2_RQSTS_ALL_DEMAND_DATA_RD_MISS	29.14
L2_RQSTS_ALL_DEMAND_DATA_RD_HIT	35.09
L2_RQSTS_ALL_DEMAND_DATA_RD	39.43
L2_TRANS_DEMAND_DATA_RD	52.43
L2_RQSTS_ALL_DEMAND_DATA_RD_MISS	56.23
L2_RQSTS_ALL_DEMAND_DATA_RD_HIT	72.32
L2_RQSTS_ALL_DEMAND_DATA_RD	35.03
L2_TRANS_DEMAND_DATA_RD	75.24
L2_RQSTS_ALL_DEMAND_DATA_RD	80.33
RXL_FLITS_G2_NCB_DATA	100
RXL_FLITS_G2_NCB_NONDATA	100
TXL_FLITS_G0_DATA	100
TXL_FLITS_G1_DRS_DATA	100
TXL_FLITS_G1_DRS_NONDATA	100
TXL_FLITS_G2_NCB_DATA	100
LSD_UOPS	42

5.1. Core and Memory Pinning

We ran two sets of experiments, one with the application pinned to the cores and the other with the application pinned to cores and memory. While the percentage errors were reduced slightly when the application is pinned to both the cores and the memory, we observed that memory pinning has no effect on *additive* PMCs but, most importantly, *non-additive* PMCs remained *non-additive* (within a tolerance of 5%).

6. Discussion

From Tab. 3 and Tab. 4 showing *potentially additive* and *non-additive* Likwid PMCs respectively, one can observe that out of a total of 151 PMCs, 43 PMCs are *non-additive*.

Table 5. List of potentially *additive* PAPI PMCs

PAPI_L1_DCM	PAPI_FUL_CCY	PAPI_L2_DCW
PAPI_L2_DCM	PAPI_BR_UCN	PAPI_L3_DCW
PAPI_CA_SHR	PAPI_BR_CN	PAPI_L3_TCR
PAPI_CA_CLN	PAPI_BR_TKN	PAPI_L2_TCW
PAPI_CA_INV	PAPI_BR_NTK	PAPI_L3_TCW
PAPI_CA_ITV	PAPI_BR_MSP	PAPI_REF_CYC
PAPI_L1_STM	PAPI_BR_PRC	PAPI_L1_TCM
PAPI_L2_LDM	PAPI_TOT_INS	PAPI_L2_TCM
PAPI_L2_STM	PAPI_L2_DCR	PAPI_BR_INS
PAPI_PRF_DM	PAPI_L3_DCR	PAPI_RES_STL
PAPI_TOT_CYC	PAPI_L2_DCA	PAPI_L3_DCA
PAPI_L2_TCA	PAPI_L2_TCR	PAPI_L3_TCA

Table 6. List of *non-additive* PAPI PMCs

Event Name	**Maximum Percentage Error (%)**
PAPI_CA_SNP	40.23
PAPI_TLB_DM	31.54
PAPI_TLB_IM	23.70
PAPI_STL_CCY	31.43
PAPI_LD_INS	32.06
PAPI_SR_INS	21.98
PAPI_LST_INS	45.87
PAPI_L1_ICM	37.28
PAPI_L2_ICM	37.50
PAPI_L2_ICH	107.12
PAPI_L2_ICA	30.65
PAPI_L3_ICA	30.2
PAPI_L2_ICR	30.65
PAPI_L3_TCM	14.54
PAPI_L3_LDM	74.68
PAPI_L1_LDM	200.82
PAPI_L3_ICR	19.48

The event *ARITH_DIVIDER_UOPS* exhibits the highest maximum percentage error of about 3075%. This event belongs to the μOPS group of Likwid PMCs responsible for gathering PMCs related to the instruction pipeline.

Several PMCs (*HA_R2_BL_CREDITS_EMPTY_HI_HA0*, *HA_R2_BL_CREDITS_EMPTY_HI_HA1*, *HA_R2_BL_CREDITS_EMPTY_HI_R2_NCS*) show maximum percentage error of about 200%. These events specifically belong to the *Home Agent (HA)* group of Likwid PMCs. HA is central unit that is responsible for providing PMCs from protocol side of memory interactions.

There are several PMCs that show maximum percentage error of about 100%. They are mainly from the *QPI* group of Likwid PMCs responsible for packetizing requests from the caching agent on the way out to the system interface.

Similarly, from Tab. 5 and Tab. 6 showing *potentially additive* and *non-additive* PAPI PMCs respectively, 17 PMCs out of a total of 53 PMCs are *non-additive*. One PMC, *PAPI_L1_LDM*, demonstrates the highest maximum percentage error of about 200%. It represents L1 load misses. Another PMC, *PAPI_L2_ICH*, demonstrates a maximum percentage error of over 100%. It represents L2 instruction cache hits.

If we increase the tolerance to about 20%, then only 8 *non-additive* Likwid PMCs will become *potentially additive*. For PAPI, only two *non-additive* PMCs will become *potentially additive*, *PAPI_L3_TCM* and *PAPI_L3_ICR*. They represent L3 cache misses and L3 instruction cache reads respectively. Increasing the tolerance to about 30% results in other 3 *non-additive* Likwid PMCs and 5 *non-additive* PAPI PMCs becoming potentially *additive*.

Thus, one can see that there are still a large number of PMCs that are *non-additive* even after increasing the tolerance to as high as 30%. Some of these PMCs have been used as key predictor variables in energy predictive models [2, 4–7, 10, 17, 20, 23, 27, 28].

To summarize, the *non-additive* PMCs that exceed a specified tolerance must be excluded from the list of PMCs to be considered as predictor variables for energy predictive modeling, because they can potentially damage the prediction accuracy of these models due to their highly non-deterministic nature. Also the list of potentially *additive* PMCs must be further tested exhaustively for more diverse applications and platforms to secure more confidence in their *additivity*.

In our future work, we would study how much the prediction error is affected due to the presence of *non-additive* PMCs in all the linear predictive energy models that we surveyed.

Conclusion

Performance events (PMCs) are now dominant predictor variables for modeling energy consumption. Considering the large set of PMCs offered by modern processors, several techniques have been devised to select the best subset of PMCs to be used for energy predictive modeling. However, the existing techniques have not considered one fundamental property of predictor variables that should have been taken into account in the first place to remove PMCs unsuitable for modeling energy. We have addressed this oversight in this paper.

We proposed a novel selection criterion for PMCs called *additivity*, which can be used to determine the subset of PMCs that can potentially be considered for reliable energy predictive modeling. It is based on the experimental observation that the energy consumption of a serial execution of two applications is the sum of energy consumptions observed for the individual execution of each application. A linear predictive energy model is consistent if and only if its predictor variables are additive in the sense that the vector of predictor variables for a serial execution of two applications is the sum of vectors for the individual execution of each application.

We studied the *additivity* of PMCs offered by two popular tools, *Likwid* and *PAPI*, using a detailed statistical experimental methodology on a modern Intel Haswell multicore server CPU. We showed that many PMCs in Likwid and PAPI are *non-additive* and that some of these PMCs are key predictor variables in energy predictive models thereby bringing into question the reliability and reported prediction accuracy of these models.

In our future work, we would classify the *non-additivity* of a PMC into application-specific and platform-specific categories. We will also look at *additivity* of PMCs offered by accelerators such as Graphical Processing Units (GPUs). For instance, Nvidia GPUs provide CUDA Profiling Tools Interface (CUPTI) that provides functions to determine around 140 PMCs. However, implementing a compound application (or kernel) from two or more base applications (kernels) is not straightforward. While CUPTI allows a continuous event collection mode, we found it is not widely supported and hence unusable presently for implementation of compound applications.

Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

References

1. Bircher, W.L., John, L.K.: Complete system power estimation using processor performance events. IEEE Transactions on Computers 61(4), 563–577 (Apr 2012), DOI: 10.1109/TC.2011.47

2. Chadha, M., Ilsche, T., Bielert, M., Nagel, W.E.: A statistical approach to power estimation for x86 processors. In: Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International. pp. 1012–1019. IEEE (2017), DOI: 10.1109/IPDPSW.2017.98

3. CUPTI: Cuda profiling tools interface (2017), https://developer.nvidia.com/cuda-profiling-tools-interface, accessed: 2017-04-10

4. Dauwe, D., Friese, R., Pasricha, S., Maciejewski, A.A., Koenig, G.A., Siegel, H.J.: Modeling the effects on power and performance from memory interference of co-located applications in multicore systems. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2014)

5. Dolz, M.F., Kunkel, J., Chasapis, K., Catalán, S.: An analytical methodology to derive power models based on hardware and software metrics. Computer Science-Research and Development 31(4), 165–174 (2016), DOI: 10.1007/s00450-015-0298-8

6. Dolz Zaragozá, M.F., Kunkel, J., Chasapis, K., Catalán Pallarés, S.: An analytical methodology to derive power models based on hardware and software metrics (2015), DOI: 10.1007/s00450-015-0298-8

7. Eidenbenz, S.J., Djidjev, H.N., Nadiga, B.T., Park, E.J.: Simulation-based and analytical models for energy use prediction. Tech. rep., Los Alamos National Laboratory (LANL) (2016)

8. Goel, B., McKee, S.A., Gioiosa, R., Singh, K., Bhadauria, M., Cesati, M.: Portable, scalable, per-core power estimation for intelligent resource management. Green Computing Conference, 2010 International (2010-08-16 2010), DOI: 10.1109/GREENCOMP.2010.5598313

9. Gschwandtner, P., Knobloch, M., Mohr, B., Pleiter, D., Fahringer, T.: Modeling CPU energy consumption of hpc applications on the IBM POWER7. In: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on. pp. 536–543. IEEE (2014), DOI: 10.1109/PDP.2014.112

10. Haj-Yihia, J., Yasin, A., Asher, Y.B., Mendelson, A.: Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems. ACM Transactions on Architecture and Code Optimization (TACO) 13(4), 56 (2016), DOI: 10.1145/3018112

11. HCL: HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter (2016), http://git.ucd.ie/hcl/hclwattsup, accessed: 2017-04-24

12. HCL: SLOPE-PMC: Towards the automation of pmcs collection for intel based multicore platforms (2017), https://git.ucd.ie/hcl/SLOPE/tree/master/SLOPE-PMC, accessed: 2017-04-24

13. Intel Optimzed HPCG: Overview of the intel optimized hpcg, https://software.intel.com/en-us/node/599524, accessed: 2017-04-24

14. IntelPCM: Intel performance counter monitor - a better way to measure cpu utilization. (2012), https://software.intel.com/en-us/articles/intel-performance-counter-monitor, accessed: 2017-04-22

15. Jarus, M., Oleksiak, A., Piontek, T., Wglarz, J.: Runtime power usage estimation of HPC servers for various classes of real-life applications. Future Generation Computer Systems 36 (2014), DOI: 10.1016/j.future.2013.07.012

16. Kadayif, I., Chinoda, T., Kandemir, M., Vijaykirsnan, N., Irwin, M.J., Sivasubramaniam, A.: vec: Virtual energy counters. In: Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. pp. 28–31. PASTE '01, ACM (2001), DOI: 10.1145/379605.379639

17. Lively, C., Wu, X., Taylor, V., Moore, S., Chang, H.C., Su, C.Y., Cameron, K.: Power-aware predictive models of hybrid (mpi/openmp) scientific applications on multicore systems. Computer Science-Research and Development 27(4), 245–253 (2012), DOI: 10.1007/s00450-011-0190-0

18. O'Brien, K., Pietri, I., Reddy, R., Lastovetsky, A., Sakellariou, R.: A survey of power and energy predictive models in HPC systems and applications. ACM Computing Surveys 50(3) (2017), DOI: 10.1145/3078811

19. PAPI: Performance application programming interface 5.5.1 (2017), http://icl.cs.utk.edu/papi/, accessed: 2017-04-24

20. Singh, K., Bhadauria, M., McKee, S.A.: Real time power estimation and thread scheduling via performance counters. ACM SIGARCH Computer Architecture News 37(2), 46–55 (2009), DOI: 10.1145/1577129.1577137

21. Song, S., Su, C., Rountree, B., Cameron, K.W.: A simplified and accurate model of power-performance efficiency on emergent GPU architectures. In: 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS). pp. 673–686. IEEE Computer Society (2013), DOI: 10.1109/IPDPS.2013.73

22. Treibig, J., Hager, G., Wellein, G.: Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on. pp. 207–216. IEEE (2010), DOI: 10.1109/ICPPW.2010.38

23. Wang, S.: Software power analysis and optimization for power-aware multicore systems. Wayne State University (2014), https://digitalcoomons.wayne.edu/oa_dissertations/933/, accessed: 2017-04-24

24. Waterland, A.: Stress. https://people.seas.harvard.edu/~apw/stress/ (2001), accessed: 2017-04-24

25. Wiki, P.: perf: Linux profiling with performance counters (2017), https://perf.wiki.kernel.org/index.php/Main_Page, accessed: 2017-04-23

26. Witkowski, M., Oleksiak, A., Piontek, T., Weglarz, J.: Practical power consumption estimation for real life HPC applications. Future Gener. Comput. Syst. 29(1) (Jan 2013), DOI: 10.1016/j.future.2012.06.003

27. Wu, X., Chang, H.C., Moore, S., Taylor, V., Su, C.Y., Terpstra, D., Lively, C., Cameron, K., Lee, C.W.: Mummi: multiple metrics modeling infrastructure for exploring performance and power modeling. In: Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery. p. 36. ACM (2013), DOI: 10.1145/2484762.2484773

28. Wu, X., Lively, C., Taylor, V., Chang, H.C., Su, C.Y., Cameron, K., Moore, S., Terpstra, D., Weaver, V.: Mummi: multiple metrics modeling infrastructure. In: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on. pp. 289–295. IEEE (2013), DOI: 10.1109/SNPD.2013.73

29. Wu, X., Taylor, V., Cook, J., Mucci, P.J.: Using Performance-Power modeling to improve energy efficiency of HPC applications. Computer 49(10), 20–29 (2016), DOI: 10.1109/MC.2016.311

DOI: 10.14529/jsfi170405

Cloud Service for Solution of Promising Problems of Nanotechnology

Marina A. Kornilina[1], *Viktoriia O. Podryga*[1], *Sergey V. Polyakov*[1,2], *Dmitry V. Puzyrkov*[1], *Mikhail V. Yakoboskiy*[1]

The paper presents the problem of creating a cloud service designed to solve promising nanotechnology problems on supercomputer systems. The motivation for creating such a service was the need to integrate ideas, knowledge and computing technologies related to this applied problem, as well as the need to involve specialists in solving problems of this type. The preliminary result of the work is a prototype of the cloud environment, implemented as a KIAM Multilogin service and an application software accessible from users virtual machines. The first applications of the service were the software packages GIMM_NANO and Flow_and_Particles, designed to solve the actual problems of nanoelectronics, laser nanotechnology, multiscale problems of applied gas dynamics. The implementation of the service took into account such aspects as support for parallel computations on the park of remote supercomputers, improving the efficiency of parallelization, very large data sets processing, visualization of supercomputer modeling results. With the help of the implemented service, it was possible to optimize the process of solving the applied problems associated with calculating the parameters of gas-dynamic flows in the microchannels of industrial spraying systems. In particular, it was possible to carry out a series of studies devoted to the analysis of gas-dynamic processes at gas-metal boundary. In these studies it was shown that in the presence of microcapillaries in a technical system, it is necessary to use direct modeling of gas dynamic processes on the basis of the first principles in the Knudsen layers, for example, using molecular dynamics methods.

Keywords: cloud service, virtualization, supercomputer modeling in nanotechnology problems, visualization.

Introduction

The present work is devoted to the development of applied cloud services intended for solving complex nanotechnology problems by computer methods. Fundamental nature and relevance of the common problem comprise the fact that at present in context of the introduction of nanotechnology in many industries there is an urgent need to combine various mathematical approaches, information and computing resources into a unified tool of computer and supercomputer modeling. The most successful way of such an association is to create relevant cloud environments and services in which each user can have access to all possible information materials, modeling programs, computing resources and industrial CAD.

The specific task is to create a cloud service using heterogeneous clusters and supercomputers for multiscale modeling of nonlinear processes in polydisperse multicomponent media used in the implementation of industrial nanotechnologies. To solve the problem, the KIAM Multilogin cloud platform has been developed and implemented in the previous two years, it allows accessing the created modeling environment. The architecture of the platform and the individual components were described in sufficient detail in [1–3]. The application part of the service was represented by the parallel program Flow_and_Particles [4–6], intended for multiscale molecular modeling of processes of interacting gas particles and the walls of microchannels of

[1]Keldysh Institute of Applied Mathematics of RAS, Moscow, Russian Federation
[2]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russian Federation

technical systems. However, in connection with the evolution of this application to the scale of the software package and the availability of other software tools which the developers had, including the GIMM_NANO [7] software, the problem of integrating such programs into large systems and/or environments appeared.

The article is organized as follows. Section 1 is devoted to describing the problems and formulations of specific tasks that need to be solved. Section 2 describes the architecture of the cloud service being developed and user scenarios of work. In Section 3, the prototype of the cloud service and its individual components are described. Section 4 illustrates the application of the developed service to the example of solving the problem of modeling gas flows in the microchannels of technical systems. Conclusion summarizes the results and determines the further development of the service.

1. Problems and Tasks

In this paper, the task was to create a cloud-based application service. The motivation for this task was the general trends in the development of software for mass use. Within the framework of cloud technologies, many problems are solved: hierarchical user authentication, the creation of a common protected information space, the work of users with personal desktops of virtual machines, migration of virtual machines in the constantly expanding and modifying field of calculators, optimal management of computing resources, convenience of developing specific applications, guaranteed secure storage of critical information, joint software development, training in work with services, and much more.

The specific task was, first of all, the development of a general concept of the applied part of the medium associated with supercomputer modeling of multiscale nonlinear processes in real technical microsystems. The complexity of this kind of problem is due to the fact that the mathematical models used contain heterogeneous descriptions of the physical processes under consideration that pertain to different scale levels. These heterogeneity and different scales give rise to a number of problems that have to be solved simultaneously.

Let us explain what has been said on a certain example. For this, let us consider the problem of calculating gas flow through a metallic channels system of a real three-dimensional geometry. In the case of a macroscale problem (when the sizes of the modeled system are much larger than the mean free path of molecules, that is, the Knudsen number is much less than 1) and simple Cartesian geometry, the description of the flow will require the use of one of the models of continuum mechanics (for example, the systems of Navier-Stokes equations for describing the gas flow, as well as the heat conduction equation for describing the heat exchange of a gas with the walls of the channel). The final initial-boundary task can be solved either with the help of ready-made software (for example, with the help of packages ANSYS CFX, StarCD, OpenFOAM, FlowVision, etc.), or by developing own solution.

In essence, in the case of simple geometry, it is necessary to use the grid finite difference/volume method (FDM/FVM) or the finite element method (FEM) in combination with the method of markers and cells on a suitable structured grid. Parallel implementation of the resulting numerical algorithm is usually not particularly complicated. However, this requires a good library of parallel algorithms for solving systems of linear and/or nonlinear algebraic equations (for example, Aztec, PETSc, etc.). The use of hybrid computing, for example, systems with central and graphics processors, may present some difficulties. However, there are already a lot of tools from NVidia, Intel and other developers.

If you need to take into account the curvilinear geometry of the system, it is necessary to use unstructured grids and specialized versions of FVM or FEM. If the geometry of the problem is rather complicated, and the degree of anisotropy of the computational domain and grid is high, then for high-precision parallel calculation a parallel grid generation, its optimal partitioning according to calculators, a special scheme for storing calculation control points and calculation results, an efficient parallel algorithms for solving algebraic problems with singular matrices will be required. In this situation, widespread software may not be suitable either because of its high cost, or because there are no reliable algorithms for solving the problem at all stages (assigning and/or importing and analyzing the geometry of the computational domain, constructing and partitioning the grid or multiple grids, forming a set of algebraic tasks, solving the algebraic problems, preservation, processing and visualization of results).

If it is necessary to take into account the more complex physics of the process, then the problem of the formation of a realistic mathematical model is added to the problems of processing geometric data. In the case under consideration, it is necessary to take into account the nonideality of the gas and the walls of the channel, the mutual influence of the gas and solid media at the boundaries. As a result, it is necessary to modify the model and methods of its numerical implementation, introducing the material coefficients of the medium, including the real equations of state for the gas, the dependences of the viscosity (shear and volume), thermal conductivity and diffusion coefficients from temperature and pressure (for both gas and material of the channel walls), special algebraic models of the boundary layer.

Specified complication of the mathematical model is impossible without knowledge of the properties of substances in a wide range of parameters. This information is not always available in full, even for well-researched materials. For new composite materials, it is practically nonexistent. Therefore, in order to carry out the basic study, additional studies are needed on the properties of the gas and solid media used in the model. At present, in addition to natural experiments, there are three main approaches for obtaining information about the properties of matter: quantum-mechanical, statistical and hybrid.

The first approach is related to ab initio methods (from the first principles), which allow obtaining detailed information about substances at the molecular and submolecular levels by solving the corresponding quantum-mechanical problems. However, this approach is the most computationally capacious and does not allow obtaining in real time an information about large particle systems (in the present situation, the size of the quantum-mechanical systems being studied amounts to several thousand structural units - molecules, atoms, electrons, protons, neutrons, etc.). Therefore, it is necessary to simplify the corresponding models, moving to hybrid schemes.

Within the framework of the statistical approach, it is possible to consider very large systems (currently the number of random variables can reach 15–20 decimal orders), but the accuracy of the data obtained in large systems first increases with the number of trials, but then decreases sharply due to increasing parasitic noise. As a result, and in the framework of this approach, we have to look for hybrid solutions.

Among the most effective hybrid approaches, we select methods for solving problems based on the Boltzmann kinetic equation or Fokker-Planck equation in conjunction with statistical Monte Carlo methods, as well as methods of classical and quantum molecular dynamics. These approaches successfully combine models from the first principles with statistical and variational calculations of moderate volume and allow with good accuracy to determine the properties of

different media and materials. Based on these approaches, databases on the properties of matter are usually formed. In particular, the NIST database (USA), as well as the Russian EPIDIF database, etc. are widely known. However, the use of such databases is possible only in a limited range of parameters and in off-line mode, when the user is forced to manually collect the necessary data. Therefore, researchers have to independently create similar databases using different packages for quantum-mechanical, statistical and molecular-dynamics calculations. Among them we note AB INITIO, GAMESS, GAUSSIAN, VASP, HyperChem, LAMMPS, GROMACS and many others.

In case of a meso- or microscale problem (when in some zones of the flow the Knudsen number approaches or even exceeds 1), an additional problem is the violation of the hypothesis of continuity of the medium. In this situation it is necessary to use mixed models that combine descriptions of continuum mechanics and alternative descriptions (meso- and microscopic). At the program level, this means, in the general calculation, using the algorithms of various kinds implemented within the framework of different libraries and/or packages. The only correct possibility of such computations is the method of splitting by physical processes. Within the framework of this method, it is possible to interface almost any computational schemes (it is only necessary to satisfy the stability conditions of the general algorithm). In the presented paper, exactly this concept is laid in the basis of cloud service that is being developed.

2. The Architecture of the Cloud Service and the Scenarios of Users Work

Let us briefly consider the architecture of the application cloud service that is being developed. It assumes the existence of a distributed computing platform hidden from users and a single scheme for accessing it via the Internet/Intranet using a specialized web portal. Standard user authentication mechanisms (login-password, login-public key, etc.) and their role differentiation (novice user, user-researcher, developer, administrator, etc.) are assumed here. Inside this cloud architecture, in addition to access servers, there are an information portal, a training portal, a research portal, a developer portal, and computing resources in the form of database servers and working computers (clusters and supercomputers).

Access servers form a decentralized system in which individual elements can be spaced territorially, but synchronized according to a certain principle. Access servers contain, in addition to the authentication service, a platform for launching and operating virtual machines (VM) of users. User virtual machines are stored in one or more databases (DBVM). They can be started by the service at any time and on any access server, but only in a single copy. Simultaneous launch of a user's virtual machine on multiple access servers is excluded. In this case, it is possible to migrate the user's virtual machine to a less-loaded access server, performed by storing all VM data and cold restart. Also, each user can have several variants of VM, distinguished by the type of OS and/or intended purpose, for example, by the role function (user, developer, administrator), as well as several saved states of a particular VM (snaphots).

The user's scenario of work starts with the procedure for registering it on the access servers as a novice user, preparing one or more virtual machines (usually selecting the VM from the repository according to the user's specific preferences), exploring the capabilities of a cloud environment on the information portal, learning to work within the framework of specific projects. After passing the initial training course, the user gets the status of a user-researcher and/or a

developer user and can start working within one or several projects. At the same time, he or she has the opportunity to initiate a new project.

The scenario of the user-researcher's work is connected with preparation and carrying out of long cycles of calculations within the framework of one or several projects. This involves all the main application services of the cloud environment. Relating to the problems of computational fluid dynamics, the following commercial or open software is necessary to use: CAD systems for specifying geometry (for example, SolidWorks, ParaSolid, etc.), mesh generators (for example, GAMBIT, TetGen, Ani3D, etc.), programs of grid partitioning by calculators (for example, METIS, ParMETIS, Jostle, etc.), specific solvers from the set of available packages (for example, ANSYS CFX, StarCD, Comsol, OpenFOAM, FemLab, etc.), sequential and parallel visualizers of the received intermediate and resulting data (for example, TecPlot , ParaView, MolDraw, ChemDraw, etc.), as well as programs for pre- and post-processing of data at all stages of calculations. Programs for pre- and post-processing are necessary for combining separate stages of computation by data. The work scenario of the user-developer of the software is associated with the use of specialized virtual machines (equipped with the necessary licensed and/or freely propagated programming tools), installing ready-made software packages on computers, as well as developing and testing new programs and services in various OS and environments.

3. Cloud Service as a Way of Integrating Technology

The above-described service architecture and users work scenarios are inherent in many cloud systems. However, in the created service variant such a standard solution is only a part of the overall implementation. More complex and relevant parts in this case are the systems of managing informational, educational and computing resources, as well as a project management system. To date, in the evolving service, in a full-scale version, there is only a system for managing computing resources and computing tasks for users. It is called KIAM Job_Control and was partially described in [2].

At this stage of the cloud project development, it allows users to perform large-scale calculations on the basis of separate parallel programs from the GIMM_NANO and Flow_and_Particles packages, and also store the results in the corresponding databases, post-process and visualize them [8].

Informational and educational resources management systems are under development and will eventually represent a distributed knowledge base on all aspects of the application service: from technologies of its realization to modeling technologies implemented within user-accessible application program packages and separate applications.

The project management system has not yet been implemented, although such systems are widely used in business and in foreign government institutions. Apparently, there is a strong non-determinism of the processes of scientific research. Therefore, within the framework of this direction it is supposed to start with the automation of application development processes.

To implement this task, there is a certain reserve obtained when creating the GIMM_NANO package. Within the framework of this project, the architecture and individual elements of the designer of parallel hybrid applications in the field of solving nanotechnology problems on the basis of ready library functions were determined. In particular, the program interface for C++, C and Fortran languages was developed, which allows assembling a hybrid MPI application from some ready-made components and a newly developed calculation part that determines the novelty and target function of the code.

The ready-made components of the application include:

- input/output functions (both sequential and distributed) of text and binary data related to the description of the geometry of the calculation area, grid parameters, physical parameters of the problem, parameters of the numerical approach, parameters of parallel computations, fields of calculated data, etc.;

- parallel generators of grids of a certain type (Cartesian, triangular, tetrahedral, hybrid, block, etc.); and

- parallel solvers of systems of linear and nonlinear equations supporting calculations on central processing units (CPU), vector processing units (VPU) and graphics processing units (GPU).

The development of the application as a whole is carried out in accordance with the concept of a hybrid parallel platform [9], the distinguishing feature of which is the desire to maximize accounting the features of the problem being solved and the architecture of a calculator with minimal differences in the code from the implementation on the CPU. Achievement of such parameters is realized due to writing of special functions and macroses, taking into account specific parameters of algorithms and capabilities of calculators. Basically, the success or failure of such an implementation depends on the possibilities of vectorization of calculations and the use of block processing of poorly structured data.

When integrating the solutions of the GIMM_NANO package based on the models of continuum mechanics with molecular dynamics applications from the Flow_and_Particles package, it was possible to implement multiscale two-level parallel algorithms and corresponding programs that allow solving new types of problems of technical gas dynamics characteristic of modern nanotechnology. An example of a solution to one such problem is presented in [10]. It is important to emphasize that rapid development of this application was accomplished through the use of technologies implemented in the GIMM_NANO package.

4. Modeling of Gas Flows in Microchannels of Technical Systems

Let us briefly consider the application of the developed service to the solution of the problem of modeling gas flows in microchannels of technical systems. The basis of parallel application developed for these purposes is a combination of a gas dynamic code and a program of direct molecular modeling [6, 11, 12].The calculated geometry is shown in Fig. 1.

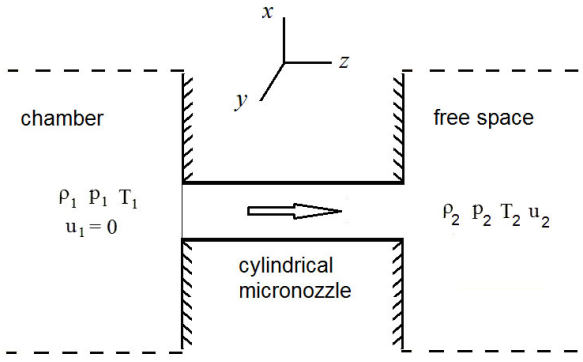


Figure 1. Calculated geometry in section $y = 0$

The essence of the problem is to calculate the gas flow from the high-pressure chamber (located on the left) through the micronozzle (located in the center) into the free space (located on the right). The complexity of the calculation is due to the fact that when the size of the nozzle decreases, the real properties of the gas and the walls of the chamber and the micronozzle turn out to be important, and not only the similarity theory is violated, but also the criterion for the continuity of the medium.

In the calculations below, we used a cylindrical micronozzle with diameter $D_0 \approx 310\ \mu m$, and length $L_0 = 6D_0 \approx 1860\ \mu m$. It connects the chamber of nitrogen and the open space of the vacuum chamber (zone of free space), which was initially filled with the same highly diluted gas. The dimensions of the computational domain were chosen as follows. The diameters of the computational parts in the chamber and in the vacuum chamber were equal to $D_1 = D_2 = 6D_0$. The length of the calculated part in the chamber was $L_1 = 10L_0$; the length of the calculated part in the vacuum chamber was equal to $L_2 = 50L_0$. Along with the true size of the investigated system, much smaller ones were considered: $D_0 \sim 1\ \mu m$.

At the initial moment, the gas is at rest: $u_1 = u_2 = 0$. In this case, it is in the chamber under standard normal conditions: $T_1 = 295.15\ K,\ p_1 = 101325\ Pa$; in the nozzle and the vacuum chamber it is at the same temperature, but at lower pressures: $T_2 = 295.15\ K,\ p_2 = \delta_0 p_1,\ \delta_0 \sim 10^{-3} \div 10^{-5}$ — pumping parameter. The nozzle on the left is blocked by a partition, which at the beginning of the calculation opens instantly.

To carry out realistic calculations, it was necessary to find out the real characteristics of the gas: the parameters of the equation of state (compressibility factor, heat capacity) and its kinetic coefficients (viscosity, thermal conductivity, diffusion, etc.). For this purpose for a sufficiently long time we were accumulating a database on the properties of nitrogen (in the temperature range from 80 to 400 K and the pressure range from 0.00001 to 1 atm), which was used as a gaseous medium (see [10, 13, 14]).

Next, a series of two-dimensional and three-dimensional calculations based on the quasi-gasdynamic (QGD) [15] model were performed [10], taking into account the real properties of nitrogen. One of the results is shown in Fig. 2 and Fig. 3. It illustrates the passage of a shock wave through a micronozzle. The figures show the results of two-dimensional modeling based on the QGD model, taking into account the real equation of state and the kinetic properties of the gas environment.

The concentration distributions for two characteristic times are represented in Fig. 2. The longitudinal velocity distributions for the same moments of time are represented in Fig. 3. The analysis of the obtained data showed that the numerical approach used reflects many real characteristics of the flow. However, some clarification is required. In particular, a more detailed study of the passage of gas through the micronozzle in case of its very small dimensions is necessary.

In connection with the foregoing, a series of numerical experiments was carried out, connected with direct molecular modeling of the processes of interacting of the gas and the micronozle walls. For this purpose, we have chosen nickel as the nozzle material, which is often used in vacuum micro- and nanotechnology. As a gas, nitrogen was still used. The conducted molecular dynamics (MD) [16] experiments comprised 3 stages.

At the first stage, the equilibrium parameters of bulk nickel were clarified, including the lattice step at 273.15 K, which amounted to $a_{Ni} \approx 0.35314\ nm$ [17, 18]. Also, the equilibrium

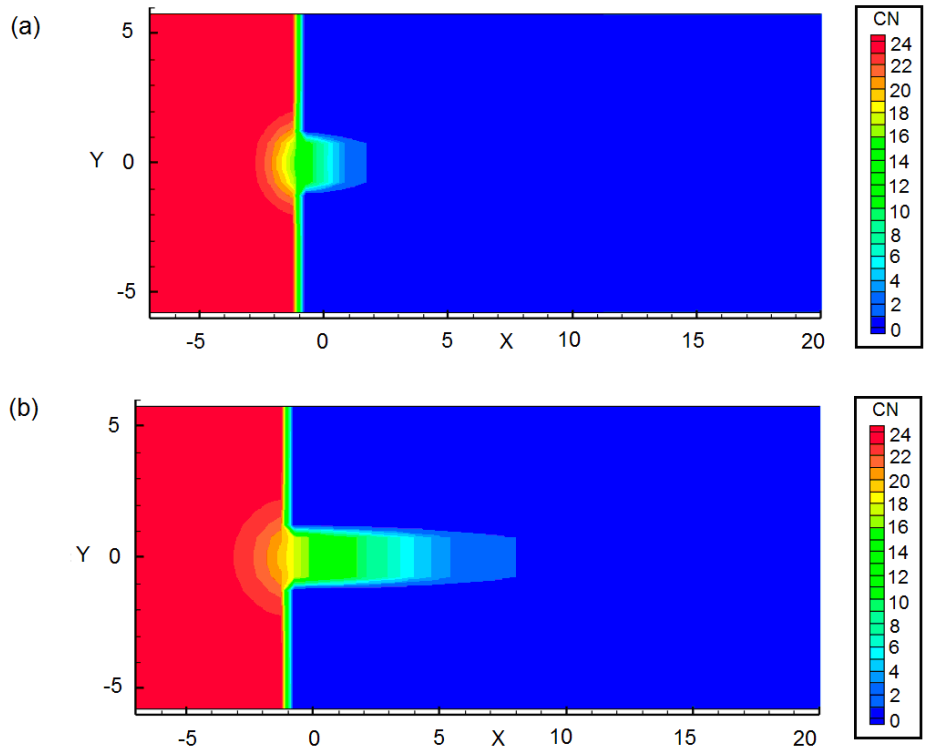


Figure 2. Distributions of the concentration of gas molecules at time points $t = 0.0553,\ 0.553\ \mu s$ (respectively, a and b)

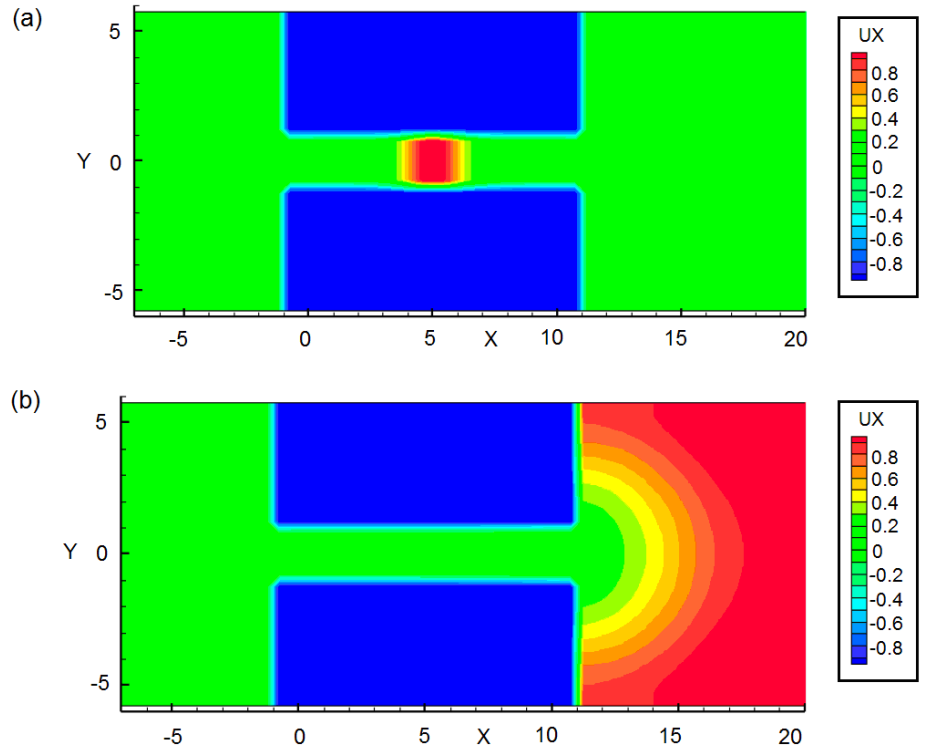


Figure 3. Distributions of the longitudinal velocity of gas molecules at time points $t = 0.0553,\ 0.553\ \mu s$ (respectively, a and b)

state of nitrogen at a given temperature was calculated, including the mean free path of nitrogen molecules $\lambda_{N_2} \approx 75\ nm$ [15].

At the second stage, the interaction of resting nitrogen with a nickel plate with dimensions 288x288x24 in units of a_{Ni} or 101.7x101.7x8.48 in nanometers [19–21]. The dimensions of the calculated region in the gas phase were 1.36x1.3640.68 in units of λ_{N_2} or 101.7x101.7x1525.6 in nanometers. The number of molecules in the considered gas-metal system was about 8.55 million particles. These calculations showed that, under normal conditions, a layer of high density nitrogen molecules is adsorbed on the nickel surface. The thickness of the layer is comparable to the triple height of the nickel crystal, that is $3a_{Ni}$. The analysis of the obtained data was carried out using the visualizer Flow_and_Particles_View [8, 22, 23] embedded in the cloud environment (Fig. 4). Note that this required processing of five thousand checkpoints of the Flow_and_Particles application with a total volume of 3.3 TB.

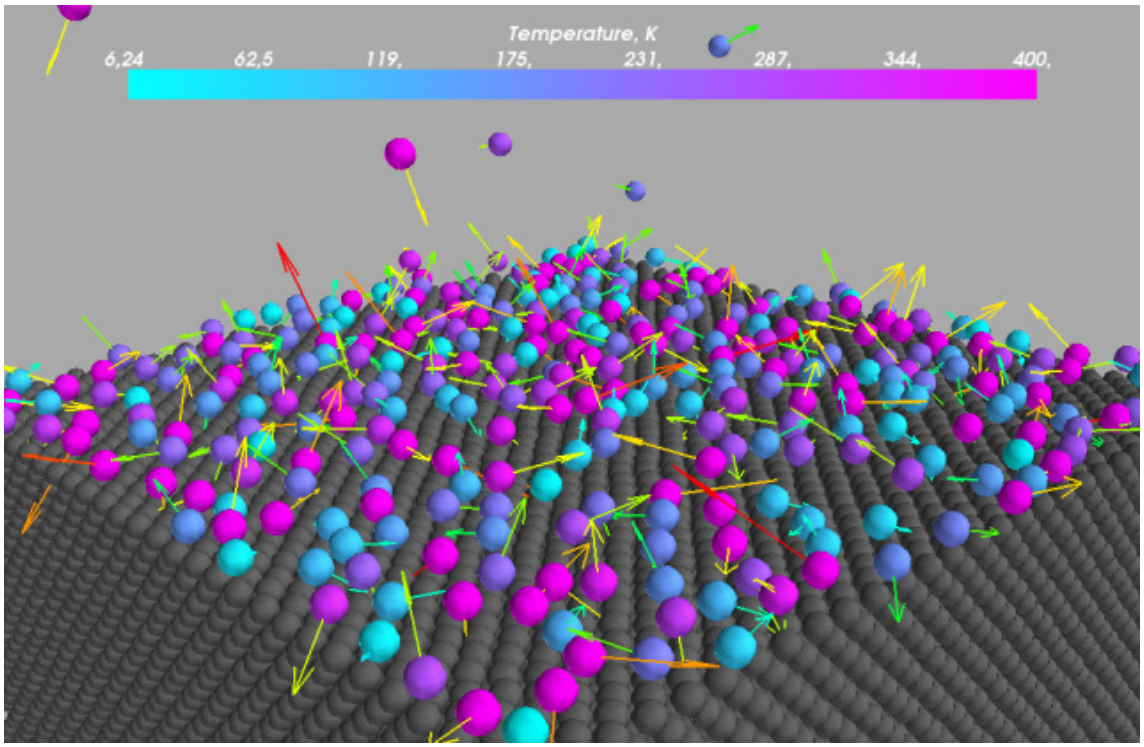


Figure 4. Visualization of effect of nitrogen adsorption on nickel surface

At the third stage, a small middle part of the micronozle was reconstructed based on the equilibrium state of the nickel plate and the nitrogen layer (Fig. 5). For reconstruction the microstructure editor KIAM MicroStructure Editor was used. The total geometry actually contained two plates of nickel (top and bottom) with layers of adsorbed nitrogen adhering to them and a free layer of nitrogen between them. The parameters of the new microsystem were: 1) the dimensions of the top and bottom plates of nickel were 288028824 in units of a_{Ni} or 1017.0x101.7x8.5 in nanometers; 2) the dimensions of the gaseous medium between the plates were 13.6x1.368.2 in units of λ_{N_2} or 1017.0x101.7x614.5 in nanometers. The number of nickel atoms in both plates is 162.57 million, the number of nitrogen molecules on the plates and between them is 42.382 million.

The new microsystem was again brought to a state of equilibrium. Then, with the help of the Langevin thermostat, the middle part (a layer of the width of $w = 1440 \cdot a_{Ni} \approx 508.5\ nm$) of the gas in it was accelerated to the velocity of 0.4 nm/ps, which is slightly higher than the speed

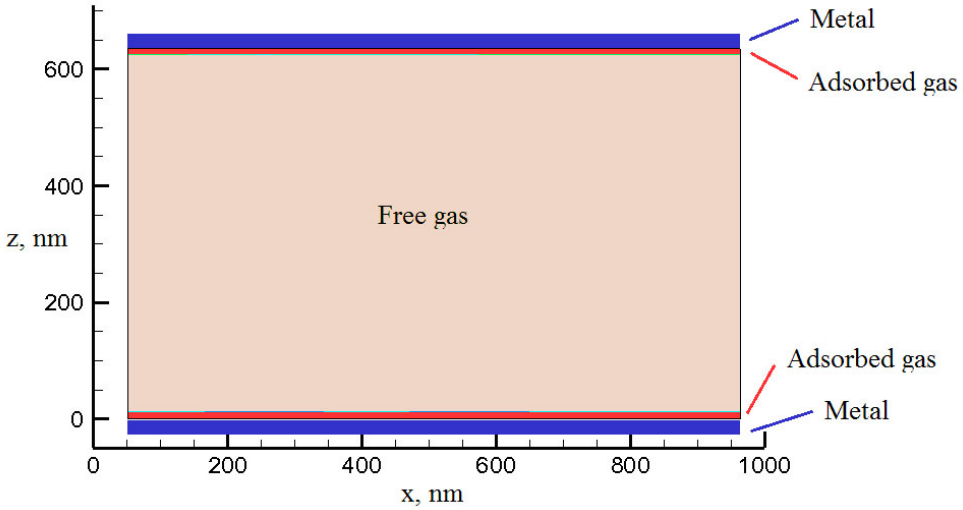


Figure 5. Calculated geometry modeling micronozle

of sound in nitrogen under normal conditions ($v_{sound} \sim 0.532\ nm/ps$). Next, the calculation with the thermostat turned off was carried out until a regular flow regime was established.

The simulation results are shown in Fig. 6, where the profiles of the longitudinal velocity component $V_x(z)$ averaged over the coordinate x are shown at different moments of time. They show first the acceleration of the gas to supersonic speed in the middle zone of the microchannel, and then the formation of a profile type of the Poiseuille profile. Dynamics with time also shows the beating of the velocity profile near the microchannel wall (Fig. 7).

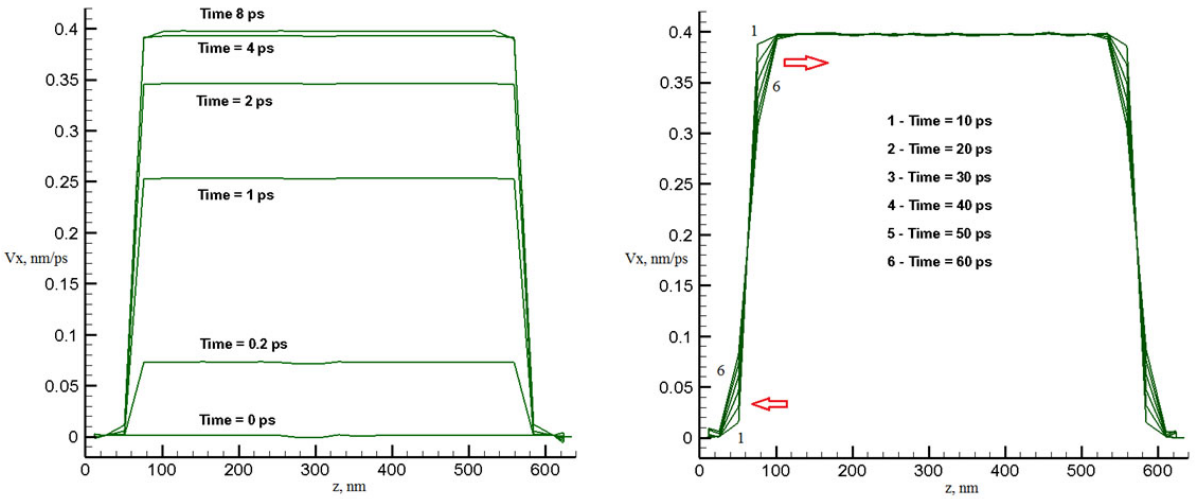


Figure 6. Evolution of the distribution of the average longitudinal velocity as a function of the coordinate z during the acceleration of the flow (on the left) and the establishment of the regular regime (right)

It is important to emphasize that these calculations would not be possible both without the use of supercomputers, and the created cloud service. In the above calculations the supercomputer K60 of Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences with a performance of 60 TFlops and the supercomputer MVS10P-PII of Joint Supercomputer Center of the Russian Academy of Sciences with a performance of 167 TFlops were used. The

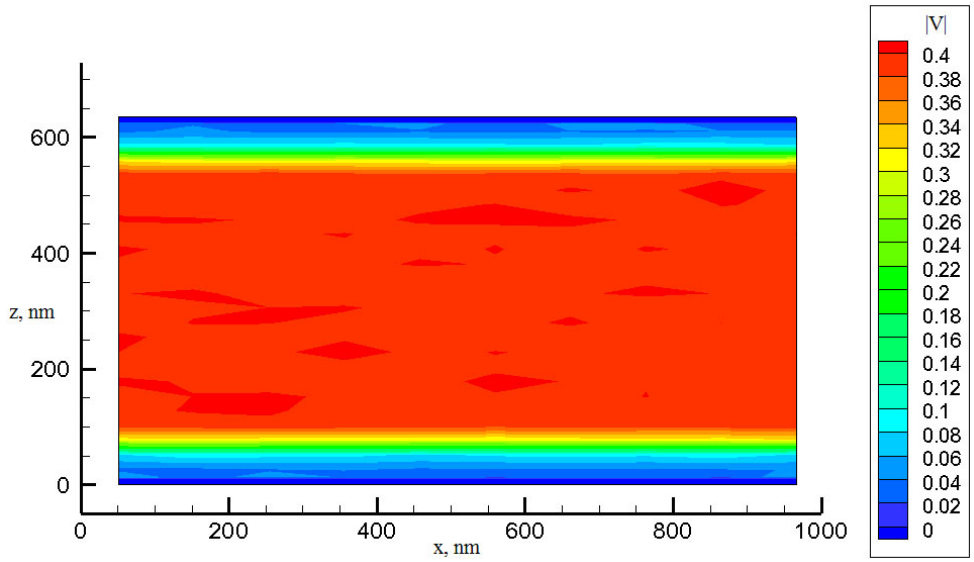


Figure 7. Distribution of the modulus of gas flow velocity in the middle part of the micronozle

developed cloud service allowed to use effectively both of these computing systems and process the total information volume of about 5 TB.

Conclusion

The paper presents a prototype of cloud service, intended for solving promising nanotechnology problems on supercomputer systems. The prototype is implemented as an access service KIAM Multilogin and application software accessible from users virtual machines. The first experience of parallel applications integration within the framework of the created service was using separate programs from the GIMM_NANO and Flow_and_Particles packages intended for solving complex problems of nanoelectronics, laser nanotechnology, and multiscale problems of technical gas dynamics. The service capabilities are demonstrated using the example of the problem of calculating the gas flow from a high-pressure chamber through a micronozzle into a free space. Specificity of the problem consists in its multiple scales and nonlinearity. To solve the problem, a two-level multiscale approach is used, realized by grid methods at the macrolevel and molecular dynamics methods at the microlevel. To achieve the necessary accuracy, the high-grid resolution and a large number of particles at the microlevel had to be used within the framework of the proposed approach. Simultaneous realization of these requirements became possible due to the use of supercomputers and the developed cloud service.

Further development of the service consists in improving the management systems of information, educational and computing resources, adding new packages and applications, creating a flexible project management system, increasing the capabilities of the visualization system.

Acknowledgments

The work was partially supported by the Academic Excellence Project of the NRNU MEPhI under contract with the Ministry of Education and Science of the Russian Federation No. 02.A03.21.0005. The work was supported by the Russian Foundation for Basic Research (projects No. 15-29-07090-ofi_m, 16-37-00417-mol_a, 17-01-00973-a).

References

1. Polyakov, S., Vyrodov, A., Puzyrkov, D., Yakobovskiy, M.: Oblachnyi servis dlia resheniia mnogomasshtabnykh zadach nanotekhnologii na superkompiuternykh sistemakh [Cloud Service for Decision of Multiscale Nanotechnology Problems on Supercomputer Systems]. Trudy ISP RAN [Proceedings ISP RAS] 27(6), 409–420 (2015). (in Russian) DOI: 10.15514/ISPRAS-2015-27(6)-26

2. Puzyrkov, D., Podryga, V., Polyakov, S., Iakobovskii, M.: KIAM_JOB_CONTROL task management environment and its application to cloud and GRID computing. In: Korenkov, V., Zaikina, T., Nechaevskiy, F. (eds.) Selected Papers of the 7th International Conference Distributed Computing and Grid-technologies in Science and Education (GRID 2016), 4-9 July 2016, Dubna, Russia. CEUR Workshop Proceedings, vol. 1787, pp. 416–422 (2017).

3. Yakobovskiy, M.V., Bondarenko, A.A., Vyrodov, A.V., Grigoriev, S.K., Kornilina, M.A., Plotnikov, A.I., Polyakov, S.V., Popov, I.V., Puzyrkov, D.V., Sukov, S.A.: Oblachnyi servis dlia resheniia mnogomasshtabnykh zadach nanotekhnologii na klasterakh i superkompiuterakh [Cloud service for solution of multiscale nanotechnology problems on clusters and supercomputers]. Izvestiia IuFU. Tekhnicheskie nauki [Izvestiya SFedU. Engineering sciences] 12, 103–114 (2016). (in Russian) DOI: 10.18522/2311-3103-2016-12-103114

4. Podryga, V.O.: Multiscale Approach to Computation of Three-Dimensional Gas Mixture Flows in Engineering Microchannels. Doklady Mathematics 94(1), 458–460 (2016). DOI: 10.1134/S1064562416040311

5. Polyakov, S., Podryga, V., Puzyrkov, D., Kudryashova, T.: Parallel Software for Simulation of Nonlinear Processes in Technical Microsystems. In: Voevodin, V., Sobolev S. (eds.) Supercomputing. Second Russian Supercomputing Days, RuSCDays 2016, 26-27 September 2016, Moscow, Russia. Communications in Computer and Information Science, vol. 687, pp. 185–198. Springer (2017). DOI: 10.1007/978-3-319-55669-7_15

6. Podryga, V.: Computational technology of multiscale modeling the gas flows in microchannels. IOP Conference Series: Materials Science and Engineering 158, 012078 (2016). DOI: 10.1088/1757-899X/158/1/012078

7. Bondarenko, A.A., Polyakov, S.V., Yakobovskiy, M.V., Kosolapov, O.A., Kononov, A.M.: Programmnyi kompleks GIMM_NANO [Software package GIMM_NANO]. In: Trudy Mezhdunarodnaia superkompiuternaia konferentsiia Nauchnyi servis v seti Internet: vse grani parallelizma [Proceedings of International Supercomputer Conference Scientific service on the Internet: all facets of parallelism], 23-28 September 2013, Novorossiysk, Russia. pp. 333–337. Moscow, Izdatelstvo MGU [Publishing MSU] (2013). (in Russian)

8. Puzyrkov, D., Podryga, V., Polyakov, S.: Parallel processing and visualization for results of molecular simulations problems. Proceedings ISP RAS 28(2), 221–242 (2016). DOI: 10.15514/ISPRAS-2016-28(2)-15

9. Polyakov, S.V., Karamzin, Yu. N., Kosolapov, O.A., Kudryashova T.A. Sukov, S.A.: Gibridnaia superkompiuternaia platforma i razrabotka prilozhenii dlia resheniia zadach mekhaniki sploshnoi sredy setochnymi metodami [Hybrid supercomputer platform and applications programming for the solution of continuous mechanics problems by grid methods]. Izvestiia

IuFU. Tekhnicheskie nauki [Izvestiya SFedU. Engineering sciences] 6(131), 105–115 (2012). (in Russian)

10. Podryga, V.O., Polyakov, S.V.: Mnogomasshtabnoe modelirovanie istecheniia gazovoi strui v vacuum [Multiscale Modeling of Gas Jet Outflow to Vacuum]. Moscow, Preprinty IPM im. M.V. Keldysha [KIAM Preprints], No. 81. (2016). (in Russian) DOI: 10.20948/prepr-2016-81

11. Podryga, V., Polyakov, S.: Parallel realization of multiscale approach for calculating the gas flows in microchannels of technical systems. In: Sokolinsky, L., Starodubov, I. (eds.) Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, PCT 2016, 29-31 March 2016, Arkhangelsk, Russia. CEUR Workshop Proceedings, vol. 1576, pp. 270–283 (2016).

12. Podryga, V., Polyakov, S.: Calculation of nitrogen flow in nickel micronozzle based on numerical approaches of gas and molecular dynamics. In: Wriggers, P. (eds.) Proceedings of the V International Conference on Particle-Based Methods. Fundamentals and Applications, PARTICLES 2017, 26-28 September 2017, Hannover, Germany. pp. 744–754. Barcelona, Spain, CIMNE (2017).

13. Podryga, V., Polyakov, S.: Correction of the Gas Flow Parameters by Molecular Dynamics. In: Onate, E. (eds.) Proceedings of the IV International Conference on Particle-Based Methods. Fundamentals and Applications, PARTICLES 2015, 28-30 September 2015, Barcelona, Spain. pp. 779–788. Barcelona, Spain, CIMNE (2015).

14. Podryga, V.O.: Calculation of Kinetic Coefficients for Real Gases on Example of Nitrogen. In: Dimov, I., Farag I., Vulkov, L. (eds.) 6th International Conference Numerical Analysis and Its Applications, NAA 2016, 15-22 June 2016, Lozenetz, Bulgaria. Lecture Notes in Computer Science, vol. 10187, pp. 542–549. Springer, Cham (2017). DOI: 10.1007/978-3-319-57099-0_61

15. Elizarova, T.G.: Quasi-Gas Dynamic Equations. Computational Fluid and Solid Mechanics, Springer (2009). DOI: 10.1007/978-3-642-00292-2

16. Haile, J.M.: Molecular Dynamics Simulations. Elementary Methods. New-York, John Wiley and Sons, Inc. (1992).

17. Podryga, V.O., Polyakov, S.V.: Molecular dynamics simulation of thermodynamic equilibrium establishment in nickel. Mathematical Models and Computer Simulations 7(5), 456–466 (2015). DOI: 10.1134/S2070048215050105

18. Podryga, V.O., Polyakov, S.V.: 3D molecular dynamic simulation of thermodynamic equilibrium problem for heated nickel. Computer Research and Modeling 7(3), 573–579 (2015).

19. Podryga, V.O., Polyakov, S.V., Puzyrkov, D.: Superkompiuternoe molekuliarnoe modelirovanie termodinamicheskogo ravnovesiia v mikrosistemakh gaz-metall [Supercomputer molecular modeling of thermodynamic equilibrium in gas-metal microsystems]. Vychislitelnye metody i programmirovanie [Numerical methods and programming] 16(1), 123–138 (2015). (in Russian)

20. Podryga, V.O., Polyakov, S.V., Zhakhovskii, V.V.: Atomisticheskii raschet perekhoda v termodinamicheskoe ravnovesie azota nad poverkhnostiu nikelia [Atomistic calculation of the

nitrogen transitions in thermodynamic equilibrium over the nickel surface]. Matematicheskoe modelirovanie [Mathematical Simulation] 27(7), 91–96 (2015). (in Russian)

21. Podryga, V.O., Karamzin, Yu.N., Kudryashova, T.A., Polyakov, S.V.: Multiscale simulation of three-dimensional unsteady gas flows in microchannels of technical systems. In: Papadrakakis, M., Papadopoulos, V., Stefanou, G., Plevris, V. (eds.) Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS Congress 2016, 510 June 2016, Crete Island, Greece. vol. 2, pp. 2331–2345 (2016).

22. Puzyrkov, D., Podryga, V., Polyakov, S.: Distributed visualization in application to the molecular dynamics simulation of equilibrium state in the gas-metal microsystems. In: Sokolinsky, L., Starodubov, I. (eds.) Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, PCT 2016, 29-31 March 2016, Arkhangelsk, Russia. CEUR Workshop Proceedings, vol. 1576, pp. 284–297 (2016).

23. Puzyrkov, D., Polyakov, S., Podryga, V.: Visualization for molecular dynamics simulation of gas and metal surface interaction. EPJ Web of Conferences 108, 02037-1–02037-6 (2016). DOI: 10.1051/epjconf/201610802037