Evaluating Performance and Scalability of Advanced Accelerator Simulations

Jungmin Lee[1], Zhiling Lan[1], J. Amundson[2], and P.Spentzouris[2]
Department of Computer Science
Illinois Institute of Technology[1]
Fermi National Accelerator Laboratory[2]

{leejung6, lan}[1]@iit.edu
{amundson, spentz}[2]@fnal.gov

Abstract

Advanced accelerator simulations have played a prominent role in the design and analysis of modern accelerators. Given that accelerator simulations are computational intensive and various high-end clusters are available for such simulations, it is imperative to study the performance and scalability of accelerator simulations on different production systems. In this paper, we examine the performance and scaling behavior of a DOE SciDAC funded accelerator simulation package called Synergia on three different TOP500 clusters including an IA32 Linux Cluster, an IA64 Linux Cluster, and a SGI Altix 3700 system. The main objective is to understand the impact of different high-end architectures and message layers on the performance of accelerator simulations. Experiments show that IA32 using single-CPU can provide the best performance and scalability, while SGI Altix and IA32 using dual-CPU do not scale past 128 and 256 CPUs respectively. Our analysis also indicates that the existing accelerator simulations have several performance bottlenecks which prevent accelerator simulations to take full advantage of the capabilities provided by teraflop and beyond systems.

1. Introduction

Particle accelerators constitute one of the most useful and complex research instruments available to scientists today. Over the past decade, advanced accelerator simulations have played a prominent role in the design and analysis of modern accelerators. The design of the next generation accelerators such as the International Linear Collider [ILC], as well as the optimization of existing accelerators such as the Fermilab Booster [DOE03], will benefit from three-dimensional high-fidelity modeling. Accelerator simulations are computational intensive, requiring massively parallel systems. In the past decades, computing capacity has been increased dramatically and various clusters with hundreds to thousands of processors have been deployed. Teraflop computers, capable of executing one trillion (10^{12}) floating point operations per second (TFlops), have emerged. Petaflop systems (10^{15}) are expected by the end of the decade. Given that the existence of powerful large-scale clusters for accelerator simulations, it is critical to study the performance and scalability of accelerator simulations on various high-end clusters, and further to identify the performance bottlenecks associated with accelerator simulations so as to take full advantage of the capabilities of teraflop and beyond systems.

This paper presents a detailed performance and scalability analysis of advanced accelerator simulations on various TOP500 clusters [T500]. In particular, we study the performance of Synergia, a state-of-the-art accelerator modeling code funded by the DOE SciDAC program [SYNE, AS02, AS03], on three different TOP500 clusters at NCSA (IA32 Linux cluster, IA64 Linux cluster, and SGI Altix 3700 system). We start with examining the low-level machine characteristics of these clusters through microbenchmarks. We then describe the porting details of Synergia on these clusters, followed by analyzing the performance of accelerator simulations from three aspects: overall performance, computational kernels, and communication calls. The goal of this paper is to understand the performance and scaling behavior of accelerator simulations on different high-end architectures and message layers and further to investigate the performance bottlenecks associated with the existing accelerator simulations.

The rest of the paper is organized as below. In Section 2, we give an overview of accelerator simulations and the Synergia package. Section 3 describes the TOP500 clusters used in this work and compares their low-level performance through microbenchmarks. Section 4 describes our porting experiences of accelerator simulations on these systems. In Section 5, we present the performance and scalability results. Finally, Section 6 summarizes the paper.

2. Background

In this section, we give an overview of advanced accelerator simulations and the Synergia package [SYNE].

2.1. Accelerator Simulations

In high-energy physics, experiments associated with accelerators have led to important discoveries about the fundamental forces of nature and the discovery of new elementary particles [AS06].

In accelerator simulation, the modeling of beam dynamics is one of the key research areas. The dynamics of charged particles in accelerators, in absence of radiation, can be divided into two categories: (1) *single-particle optics*, which are due to the interaction of the beam particles with the electromagnetic fields produced by the magnets and the RF cavities of the accelerator and (2) *collective beam effects*, which are due to the interaction of beam particles with fields produced by the beam itself or by other beams in each vicinity [AS03]. Modeling of single-particle dynamics has been extensively studied, while modeling of collective beam effects is a less mature and much more computationally intensive endeavor, especially in the case of very intense charged beams. These collective effects often result in the development of beam halo (particles falling outside the desired beam phase space boundaries) that eventually causes beam losses and irradiation of the accelerator complex. In addition, such effects result in the dilution of the beam phase pace causing a decrease in accelerator intensity and collider luminosity. In order to accurately modeling realistic conditions of accelerator operation, the collective beam effects must be considered [AS06].

The most widely used method to model collective beam effects is the Particle-In-Cell (PIC) method [QR+00]. In this approach the beam distribution is represented by a set of macro-particles that evolve according to the single particle equations of motion; the collective effects (beam-generated electromagnetic fields) are calculated by solving the Poisson-Vlasov equation at every simulation step. Three operations are performed during the simulation at each step: first, the particles are deposited on a spatial grid; second, the fields are calculated on the grid; third and final, the fields are interpolated back to the particles and used to push the particles. A large number of macro-particles (order a few million or more) are needed to obtain the required simulation accuracy, thus the need of using of massively parallel computers.

2.2. Synergia Code

Synergia, funded by the DOE SciDAC Accelerator Science and Technology Project, provides state-of-the-art simulations of linear and circular accelerators with a fully three-dimensional treatment of space charge. It is a parallel three dimensional modeling code for multi-particle effects integrating existing beam dynamics modeling tools. It has been used to model space charge effects in the Fermilab Booster and the results are in good agreement with Booster beam study experiments. The implementation is fully parallelized.

Synergia uses IMPACT [QR+00] for its parallel simulation of the propagation of particles, the modeling of RF cavities and, most important, parallel calculations of space-charge effects. IMPACT is a 3D parallel Particle-In-Cell (PIC) code for modeling high-intensity beams in RF accelerators. For the modeling of single-particle optics, Synergia uses the mxyzptlk/beamline libraries [LM91], which can perform a wide range of accelerator physics computations. Synergia integrates the above packages using a framework written in C++ and Python. Synergia is designed to be a general-purpose framework with an interface that is accessible to accelerator physicists who are not experts in simulation. Figure1 gives an overview of Synergia. Figure 1 gives an overview of Synergia.

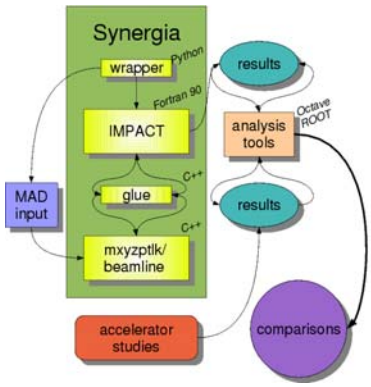


Figure 1: Overview of Synergia

3. TOP500 Clusters

In this section, we first describe main features of the TOP500 clusters used in this work, and then evaluate their low-level performance through two well-known microbenchmarks.

3.1. Overview

Three different types of TOP500 clusters at NCSA are used in this work: IA32 Linux cluster, IA64 Linux cluster, which is a part of TeraGrid [TG], and SGI Altix 3700 system.

We choose these systems for several reasons. First, they are widely used in the area of scientific computing. Secondly, they represent different high performance

computing environments. Lastly, they have different configurations of hardware, compiler, operating systems, MPI and other software environments. This can provide us with a fair evaluation of accelerator simulations.

Table 1 summarizes the overall hardware and software differences among IA32, IA64, and SGI Altix.

Component	IA32	IA64	SGI Altix
TOP 500 ranking (June 2005)	28th	49th	60th
Peak Performance	15300 GFlops	10259 Gflops	6553 Gflops
Architecture	Linux Cluster	Linux Cluster	Distributed Shared Memory (ccNUMA)
Number of nodes/CPUs	1450	887	1024
Processor	Intel Xeon 3.2 GHz (32-bit) 8 KB L1 512 KB L2 1 MB L3	Intel® Itanium® 2, 1.3 GHz 32 KB L1 256 KB L2 Integrated 3 MB L3	Intel® Itanium® 2, 1.5 GHz 32 KB L1 256 KB L2 6MB L3 cache
Network Interconnect	Giga Ethernet & Myrinet 2000	Giga. Ethernet & Myrinet2000	SGI NUMAlink3
Compiler	Intel: Fortran77/90/95 C C++ & GNU: Fortran77 C C++	Intel: Fortran77/90/95 C C++ & GNU: Fortran77 C C++	Intel 8.1: Fortran77/90/95 C C++ & GNU : Fortran77 C C++
Operating System	Linux 2.4.20 (Red Hat 9.0)	SUSE Linux SLES8 / 2.4 MP	SGI ProPack 3.4 Kernel : 2.4.21-sgi304
MPI	MPICH-GM, MPICH-G2, MPICH-VMI	MPICH-GM, MPICH-G2, MPICH-VMI	SHMEM

Table 1: Architectural Specifications on IA32/64 and SGI Altix at NCSA

As shown in the table, these clusters are different in terms of architecture, processor type, and network connections. Both IA32 and IA64 are distributed address space machines, while SGI Altix is a ccNUMA (Cache Coherent Non-Uniform Memory Access) machine. Both IA64 and SGI Altix use Intel Itanium2 processors that are Intel's 64-bit processors, while IA32 uses 32-bit Intel Xeon processors. The major difference between Xeon processors and Itanium processors is the bitness of integer registers. Also, Xeon processors have extremely high clock rates, while Itanium processors have very high floating-point speed at relatively lower clock rates. Both Gigabit Ethernet and Myrinet interconnections are supported in IA32 and IA64, SGI Altix uses SGI NUMA link3 as its network interconnection.

3.2. Microbenchmark Performance Results

We first evaluate the low-level machine characteristics of these clusters through the microbenchmark STREAM [STREAM] and Pallas MPI Benchmark (PMB) [PMB]. The STREAM benchmark is a synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels. PMB is a widely used MPI benchmark that is used to compare the performance of various MPI functions.

P	IA32	IA64	SGI Altix
2	1845.5902	652.5984	767.5930
4	1836.9757	651.3316	768.5776
8	1835.0065	652.0573	765.6838
16	1833.5297	653.3014	766.2724
32	1835.7693	654.7270	768.1963
64	1858.8730	651.9432	765.8440
128	1836.5511	652.2094	766.1733
256	1854.5013	653.7023	768.9856
512	1849.7094	652.3087	-----

Table 2: Per-processor STREAM triad performance (in MB/s) for unit stride.

Table 2 presents the performance results comparing these clusters using STREAM. Here, the data represent the asymptotic unit-stride memory bandwidth behaviour of the triad summation: $a(i) = b(i) + s \times c(i)$. The number of processors is varying from 2 to 512 processors. Note that we were not able to use 512 processors on SGI Altix due to the limitation of maximum number of nodes allocated to us. As shown in the table, IA32 cluster has better memory bandwidth as compared to IA64 and SGI Altix. The average bandwidth of IA32 is about 2.8 (2.4) times larger than that of IA64 (SGI Altix). The memory bandwidth of SGI Altix is about 1.2 times better than IA64.

Now we move to examine the message passing performance of these clusters. According to our analysis, four MPI calls are frequently used in Synergia (MPI_Send/Recv, MPI_Barrier, MPI_Allreduce and MPI_Reduce). Therefore, we study the performance of these MPI calls by using PMB.

P	8192 Bytes			2097152 Bytes		
	IA32	*IA64*	*Altix*	*IA32*	*IA64*	*SGI Altix*
2	269.73	1207.12	3868.48	457.61	1176.13	11416.75
4	258.27	337.54	1283.72	443.99	399.71	6527.79
8	242.06	337.28	1612.81	413.01	394.36	7215.23
16	212.47	335.06	1507.43	325.50	351.12	7092.17
32	224.39	215.90	1483.63	317.24	218.08	3659.94
64	230.73	227.86	1419.69	289.51	192.36	6509.73
128	189.83	141.80	1373.77	297.43	150.61	6122.13
256	212.58	112.36	1165.95	197.44	119.13	2741.87
512	170.02	114.11	----	182.66	116.05	----

Table 3: MPI Send/Recv Performance (in MB/s)

Table 3 shows the bandwidth results of MPI Send/Recv with varying message sizes (from 8192 bytes to 2097152 bytes) and varying numbers of processors (from 2 to 512).

Due to the space limit, we only show two message sizes in the table. As is shown in the table, SGI Altix has much better MPI Send/Rend performance, achieving more than 16.8 (11.5) times better bandwidth than that on IA64 (IA32). IA64 has better performance as compared to IA32 when the number of processors is small, otherwise IA32 has better performance.

Table 4 summarizes the overhead of MPI_Barrier on these clusters with varying numbers of processors (from 2 to 128). As shown in the table, SGI Altix introduces the minimal overhead among three clusters, achieving more than 12.9 (12.6) times better performance than that on IA32 (IA64). In general, IA64 has better performance as compared to IA32.

P	IA32	IA64	SGI Altix
2	9.41	2.71	1.07
4	19.77	25.04	2.21
8	32.74	46.40	3.30
16	70.47	69.61	4.80
32	136.75	128.34	6.52
64	81.35	122.97	8.32
128	222.76	143.42	12.82

Table 4: MPI Synchronization Overhead (in μsec)

Table 5 summarizes the performance of MPI_Allreduce on these systems with varying number of message sizes and varying numbers of processors. There is no big difference between IA32 and IA64, but the performance on SGI Altix is 4.3 (1.9) times worse than that on IA32 (IA64).

P	8192 Bytes			2097152 Bytes		
	IA32	*IA64*	*Altix*	*IA32*	*IA64*	*Altix*
2	73.3	46.6	33.6	21631.1	17455.1	6311.0
4	134.7	149.5	89.6	32694.3	59509.5	18016.7
8	225.0	256.1	120.5	64882.6	92230.0	25296.7
16	410.7	374.9	171.7	116397.1	103570.3	35624.6
32	550.1	605.8	222.4	158201.5	210120.4	46536.5
64	931.2	828.3	292.2	245818.3	244520.2	61750.3
128	1700.9	2045.6	340.9	469400.0	422024.4	64533.9
256	2803.7	1621.2	646.0	717508.6	474271.0	75333.3
512	3087.1	2029.2	----	673317.4	613863.4	----

Table 5: MPI_Allreduce Performance (in MB/s)

P	8192 Bytes			2097152 Bytes		
	IA32	*IA64*	*Altix*	*IA32*	*IA64*	*SGI Altix*
2	56.4	14.7	23.5	20288.4	8946.6	5426.2
4	120.5	105.4	57.6	32230.67	27753.1	13703.3
8	187.8	173.5	89.8	44671.9	41479.4	21171.7
16	274.4	244.4	119.5	60280.5	54001.6	27675.9
32	335.1	321.4	152.9	69317.3	69950.1	35667.1
64	396.3	384.3	185.1	79270.5	82474.2	40725.7
128	476.2	457.3	213.8	92427.6	101971.5	47014.6
256	553.5	528.7	414.1	107838.6	109422.9	54155.5
512	695.6	633.8	----	114945.0	120904.5	----

Table 6: MPI_Reduce Performance (in MB/s)

Table 6 shows the performance data of MPI_Reduce on these systems. In general, IA32 achieves the best performance, with 2.4 (1.1) times better than SGI Altix (IA64).

4. Porting Details

Synergia is a hybrid code integrating several existing packages written in multiple programming languages. While using multiple components provides a powerful accelerator simulation package, it also creates a configuration problem. Multiple language issues are particularly problematic because calling conventions vary from platform to platform [AS06]. Porting a hybrid code can be a complicated task, which is further complicated by the diverse set of programming environments provided on these systems.

We first installed Synergia on IA32 Linux Cluster and did not encounter any problems. To link with MPI libraries, we used MPICH-GM (version 1.2.5.2). The code was compiled with Intel compilers.

We then installed Synergia on IA64 Linux Cluster. The major problem we had is that Synergia was initially designed for 32-bit systems and IA64 uses different length of integer. To solve this problem, we deleted the *"long"* data type from all the files in Synergia. We also tried the GNU compiler, but it did not work with Synergia on IA64.

On SGI Altix, initially we planned to use MPICH-GM, the same version as on the other two clusters, but it did not work with Synergia. Hence, we decided to use SGI SHMEM MPI version with the specific SGI MPI library, libmpi.a.

Table 7 summarizes the compilation details of Synergia on IA32, IA64, and SGI Altix.

Compiler/MPI		**IA32/64**	**SGI Altix**
C++	*Compiler*	icpc	Icpc
	Flags	-cxxlib-icc –static	-cxxlib-icc –static
	Libraries	-limf -lcprts	-limf -lcprts
FORTRAN 90	*Compiler*	Ifort	Ifort
	Flags	-g -O2 -inline_debug_info –cm -w95	-g -O2 -inline_debug_info -cm -w95
	Libraries	libifport.a libifcore.a	libifport.a libifcore.a
MPI	*Version of MPI*	mpich-1252	SHMEM
	Libraries	libfmpich.a, libmpichf90.a, libmpich.a, libpmpich.a	libmpi.a

Table 7: Compiler Options on IA32, IA64, and Altix

5. Performance and Scalability Analysis

Now we move to evaluate and analyze the performance of Synergia on IA-32, IA-64, and SGI Altix. The experimental data are divided into three categories:

overall performance, computational kernels, and communication calls. Since the nodes on IA32 and IA64 have dual CPUs, there are two different execution environments: either using one CPU per node or using both CPUs per node. On SGI Altix, only dual-CPU configuration is allowed. Therefore, we investigate the performance of Synergia under five different scenarios: IA32 with single-CPU, IA32 with dual-CPU, IA64 with single-CPU, IA64 with dual-CPU, and SGI Altix with dual-CPU. We analyze the performance of Synergia with number of processors ranging from 2 to 512, except for SGI Altix on which we only have the access to 256 processors.

5.1. Execution Time

Table 8 and Figure 2 summarize the total execution times on these systems. Here, the columns under "single CPU" represent the performance data using a single CPU per node, while the columns under "dual CPU" represent the performance data using two CPUs per node. It is shown that the total execution times on IA32 are, in average, 2.5 (2.0) times faster than those on IA64 (SGI Altix). In other words, IA32 achieves the best performance, followed by SGI Altix and then IA64.

	IA32 (single CPU)	*IA32 (dual CPU)*	*IA64 (single CPU)*	*IA64 (dual CPU)*	*SGI Altix*
2	7052	6703	16784	19064	12648
4	3495	3471	9633	8640	7105
8	2030	1832	4997	5020	3288
16	1045	969	2388	2378	2065
32	508	512	1208	1274	932
64	300	336	639	815	554
128	196	207	562	563	468
256	156	168	469	470	733
512	133	182	460	460	----

Table 8: Execution Time on IA32, IA64, and Altix

By looking into the data shown in Table 8, we notice that when the number of processors is smaller or equal to 16, the dual-CPU configuration is faster than the single-CPU configuration. When the number of processors is larger than 16, the single-CPU configuration general provides better performance as compared to the dual-CPU configuration.

We then study the speedup achieved on these systems as it can provide us more clear information of the scalability of Synergia on these systems. Figure 3 illustrates the speedup obtained on these systems. Here, speedup is defined as $\frac{(2\times T_2)}{T_p}$, where T_2 and T_p denote the total execution time on two and P processors respectively. When the number of processors is larger than 128, we notice very different scalability behavior on these systems: (1) on IA64 with single- and dual-CPU, the speedup lines keep increasing with much slower speed; (2) on IA32 with single-CPU, the speedup line is the most satisfying one even when the number of processors reaches 512 processors; (3) on IA32 with dual-CPU, the speedup line suddenly decreases when the number of processors is increased from 256 to 512; (4) the worst case is on SGI Altix, in which the speedup starts to decrease when the number of processors reaches 128 processors.

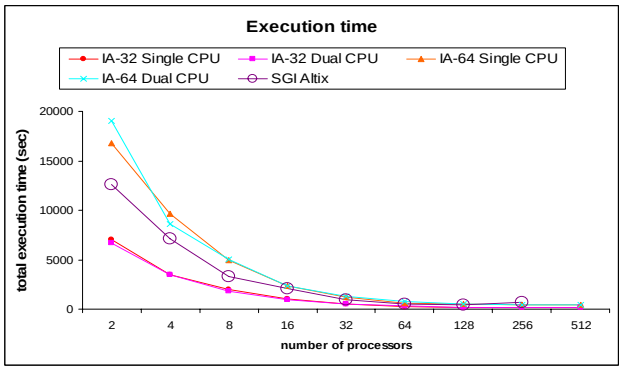


Figure 2: Execution Time on IA32, IA64 and Altix

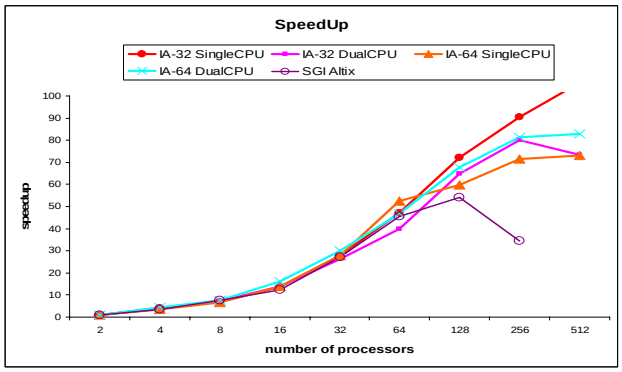


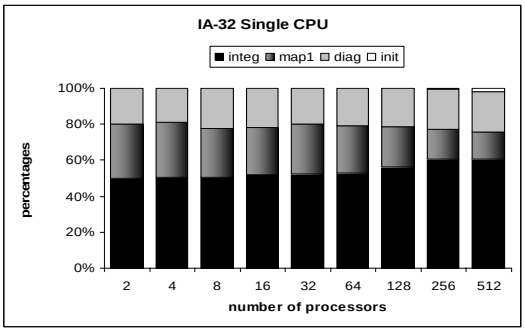
Figure 3: Speedup on IA32, IA64, and SGI Altix

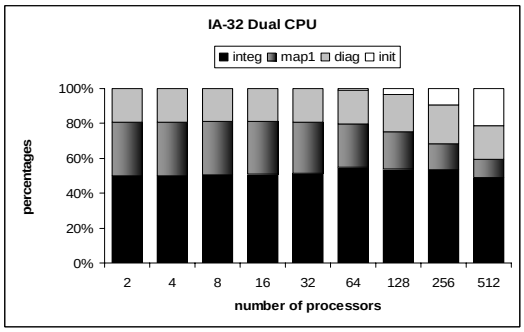
5.2. Computational Kernels

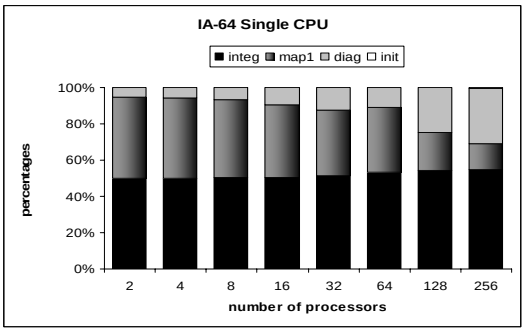
Our analysis shows that there are four major computational kernels in Synergia, which are described in Table 9.

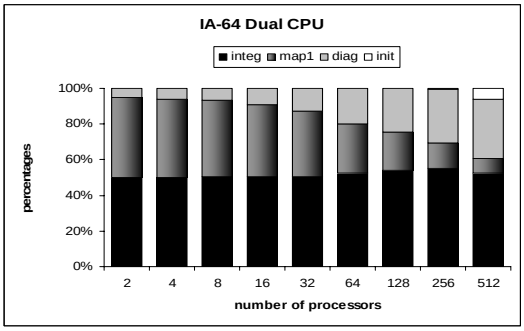
Kernel	Description
Integ	Start looping through beam line elements and numerical segments in each beam element using two-step simplistic integrations, and run advancing particles applying the space charge.
Map1	Drift beam half step using linear map for external field, nonlinear Lorenz integrator, and nonlinear Lorenz integrator for halo study.
Diag	x,y, and z emittances
Init	Initialize objects and parameters

Table 9: Kernel Description









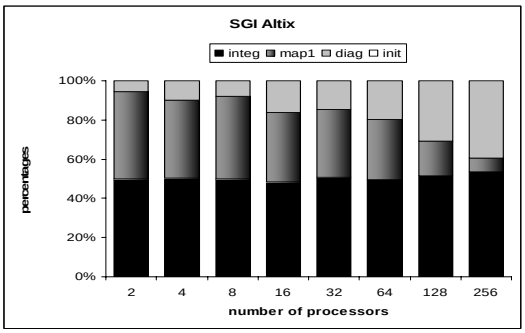


Figure 4: Percentages of Computational Kernels

Figure 4 illustrates the relative percentages of these kernels with varying numbers of processors on IA32, IA64, and SGI Altix.

The figure also shows that on IA32 with dual-CPU, the percentage of *"init"* is increasing dramatically between 256 and 512 processors. By looking into the code, we find out that the *"init"* kernel three MPI_Barrier calls. We believe it is a major performance bottleneck on IA32 with dual-CPU.

5.3. Communication Calls

To analyze the communication behavior of Synergia, two performance tools (FPMPI and mpiP) are used in our work. FPMPI provides a detailed summary of MPI calls used in a parallel program [FPMPI]. We were able to use this tool with Synergia on IA32, but failed on IA64 and SGI Altix due to the MPI_FINALIZE problem on these systems. Hence, we decided to use mpiP on IA64 and SGI Altix. Similar to FPMPI, mpiP provides profiling information of various MPI calls used in MPI applications [mpiP].

P	*IA32 (single CPU)*	*IA32 (dual CPU)*	*IA64 (single CPU)*	*IA64 (dual CPU)*	*SGI Altix*
2	0	0	88.87%	78.02%	89.29%
4	0	0	80.81%	80.69%	96.34%
8	0	0	81.83%	82.51%	82.64%
16	0	0	81.54%	82.09%	92.83%
32	0	0	85.54%	82.07%	94.45%
64	0	0.05%	76.18%	87.29%	94.12%
128	0	0.13%	85.90%	85.78%	89.63%
256	0.1%	0.20%	86.34%	85.72%	76.25%
512	0	0.62%	----	79.10%	----

Table 10: MPI_Allreduce Percentage

P	*IA32 (single CPU)*	*IA32 (dual CPU)*	*IA64 (single CPU)*	*IA64 (dual CPU)*	*SGI Altix*
2	99.7%	99.8%	0.35%	0.47%	1.36%
4	98.9%	99.4%	2.44%	6.55%	0.77%
8	97.1%	97.9%	3.34%	4.02%	8.27%
16	93.8%	96.3%	5.01%	5.19%	3.17%
32	91.2%	93.7%	1.73%	5.86%	1.03%
64	84.3%	79.6%	1.78%	1.03%	0.91%
128	77.45	74.8%	0.77%	0.76%	0.84%
256	62%	63.2%	0.61%	0.54%	12.56%
512	58.7%	48.3%	----	0.33%	----

Table 11: MPI_Reduce Percentage

Table 10 and Table 11 show the relative percentage of MPI_Allreduce and MPI_Reduce on these clusters. Here is the cost percentage. As we can see, MPI_Allreduce is the most dominant call on IA64 and SGI Altix, while MPI_Reduce is the most dominant call on IA32. MPI_Allreduce is mainly called in the code for the beam and particle distribution. Hence, we suspect that the poor performance on IA64 and SGI Altix is caused by load imbalance. MPI_Reduce is mianly called to print out

particle information. The large amount of MPI_Reduce cost on IA32 suggests that there is a parallel I/O problem on this system.

Table 12 shows the relative percentage of MPI_Send/Recv on these clusters. On IA32 with both single-CPU and dual-CPU, the percentage of MPI_Send/Recv keeps increasing. On IA64 with both single-CPU and dual-CPU, the percentage is more consistent no matter how many processors are used. The percentage on SGI Altix is the smallest, which only costs about 3%-8%. According to our PMB analysis, SGI Altix has significantly better performance on point-to-point MPI calls. In other words, the result shown in Table 12 matches the PMB result.

P	*IA32 (single CPU)*	*IA32 (dual CPU)*	*IA64 (single CPU)*	*IA64 (dual CPU)*	*SGI Altix*
2	0.4%	0.3%	7.85%	13.38%	6.57%
4	1.4%	1.4%	12.77%	9.71%	2.48%
8	3.1%	4.7%	12.73%	11.56%	8.25%
16	6.5%	8.3%	12.35%	11.66%	2.59%
32	12.0%	12.1%	12.09%	11.45%	4.16%
64	24.8%	30.3%	21.12%	11.19%	4.71%
128	36.5%	37.1%	12.81%	13.01%	4.33%
256	48.2%	46.3%	12.49%	13.03%	3.55%
512	53.5%	46.4%	----	11.72%	----

Table 12: MPI_Send/Recv Percentage

Table 13 lists the relative percentage of MPI_Barrier on these clusters. It shows that with accelerator simulation running on IA32, more time is spent on MPI_Barrier as compared to that on IA64 and SGI Altix. The smallest cost is obtained on SGI Altix. Again, this matches our PMB results shown in Section 3, that is, in terms of MPI_Barrier, IA32 has the worst performance while SGI Altix has the best performance.

P	*IA32 (single CPU)*	*IA32 (dual CPU)*	*IA64 (single CPU)*	*IA64 (dual CPU)*	*SGI Altix*
2	0.002%	0.001%	0.06%	0.05%	0.04%
4	0.01%	0.01%	0.09%	0.08%	0.02%
8	0.03%	0.04%	0.11%	0.11%	0.07%
16	0.06%	0.07%	0.12%	0.11%	0.16%
32	0.18%	0.12%	0.14%	0.12%	0.04%
64	0.27%	0.32%	0.33%	0.14%	0.05%
128	0.41%	0.40%	0.27%	0.15%	0.08%
256	0.53%	0.49%	0.27%	0.27%	0.09%
512	1.28%	0.91%	----	0.21%	----

Table 13: MPI_Barrier Percentage

5.4. Analysis Results

Based on the results presented above, we conclude that:

- In terms of the overall execution time, IA32 has the best performance followed by SGI Altix and IA64 has the worst performance. Also, with accelerator simulations on IA32 and IA64, when the number of processors is larger than 16, the single-CPU configuration outperforms the dual-CPU configuration (see Figure 2).
- In terms of scalability, accelerator simulations achieve the best scalability on IA32 with single-CPU, while the worst scalability is given on SGI Altix (see Figure 3).
- The distribution of particles among the processing nodes may become unbalanced, which is a major performance bottleneck.
- Global communication, e.g. MPI_Allreduce and MPI_Reduce, is another major performance bottleneck.
- MPI synchronization, e.g. MPI_Barrier, is one of the major performance bottlenecks on IA32 with dual-CPU.

6. Conclusion

In this paper, we studied the performance and scaling behavior advanced accelerator simulations on three TOP500 clusters (IA32, IA64, and SGI Altix). More specifically, we investigated the performance of accelerator simulations from three aspects: the overall execution time, the computational kernels, and the communication calls. Our analysis show that different scaling patterns are achieved on these systems and there are several performance bottlenecks associated with the existing accelerator simulations. In summary, the paper makes the following contributions:

- Present our experience of porting Synergia on three TOP500 clusters with different architectures and configurations;
- Study the low-level performance of these clusters through microbenchmarks;
- *Investigate the performance and scalability of Synergia on three different TOP500 clusters which are widely used in scientific computing.*

Acknowledgement

We would like to acknowledge the National Center for Supercomputing Applications (NCSA) and the TeraGrid for the use of their machines.

References:

[AS02] J.Amundson and P.Spentzouris. “Synergia: A hybrid, parallel 3d space charge code with circular machine modeling capabilities”, *Proc. of ICAP2002*, 2002.

[AS03] J.Amundson and P.Spentzouris. “Synergia: a hybrid, parallel beam dynamics code with 3D space charge”, FERMILAB-CONF-03-126-E, Jul 2003. Presented at Particle Accelerator Conference (PAC 03), Portland, Oregon, 12-16 May 2003.

[AS05] J. Amundson, P. Spentzouris. “Space charge experiments and simulation in the Fermilab Booster”, FERMILAB proceedings of Particle Accelerator Conference (PAC 05), May 2005.

[AS06] J. Amundson, P. Spentzouris. Synergia: An Accelerator Modeling Tool with 3-D Space Charge. Journal of Computational Physics, Volume 211, Issue 1, 1 January 2006, Pages 229-248.

[DOE03] DoE Review 2003. http://www-bd.fnal.gov/doereview03/

[FPMPI] FPMPI. http://www-unix.mcs.anl.gov/fpmpi/WWW/

[IA32] IA32Architecture. http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/XeonCluster/

[IA64] IA64 Architecture. http://teragrid.ncsa.uiuc.edu/TechSummary/index.html

[ILC] International Linear Collider. http://www.interactions.org/linearcollider/index.html

[JM+90] F. Jones, G. H. Mackenzie, and H. Schonauer, “Particle Accelerators”, vol. 31, 199 (1990).

[LD03] Z.Lan and P.Deshikachar. “Performance analysis of a large-scale cosmology application on three cluster systems”, *Proc. of IEEE Cluster 2003*, 2003.

[LM91] L.Michelotti, FERMILAB-CONF-91-159 and FERMILAB-FN-535-REV

[mpiP] mpiP. http://www.llnl.gov/CASC/mpip/

[PMB] Pallas MPI Benchmarks. http://www.pallas.com/e/products/pmb

[QR+00] J. Qiang, R. D. Ryne, S. Habib and V. Decyk, J. Comp. Phys. 163, 434 (2000).

[SA3000] SGI Altix 3000. http://www.sgi.com/products/servers/altix/

[STREAM] STREAM: Sustainable memory bandwidth in high performance computers. http://www.cs.virginia.edu/stream/ref.html

[Sun02] X.H. Sun, ”Scalability Versus Execution Time in Scalable Systems”, *Journal of Parallel and Distributed Computing*, Vol. 62, No. 2, pp. 173-192, Feb 2002.

[SYNE] P.Spentzouris and J.AMundson. Synergia Website. http://cepa.fnal.gov/psm/aas/Synergia.html.

[T500] Top 500 ranking. http://www.top500.org

[TG] http://www.teragrid.org