Parallel Computing 37 (2011) 550–561

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Adapting wave-front algorithms to efficiently utilize systems with deep communication hierarchies

Darren J. Kerbyson [a,*], Michael Lang [b], Scott Pakin [b]

[a] Fundamentals of Computational Sciences, Pacific Northwest National Laboratory, WA 99353, USA
[b] Computer, Computational, and Statistical Sciences, Los Alamos National Laboratory, NM 87544, USA

ARTICLE INFO

Article history:
Available online 3 March 2011

Keywords:
High performance computing
Hybrid systems
Performance analysis
Performance modeling
Programming models

ABSTRACT

Large-scale systems increasingly exhibit a differential between intra-chip and inter-chip communication performance especially in hybrid systems using accelerators. Processor-cores on the same socket are able to communicate at lower latencies, and with higher bandwidths, than cores on different sockets either within the same node or between nodes. A key challenge is to efficiently use this communication hierarchy and hence optimize performance. We consider here the class of applications that contains wave-front processing. In these applications data can only be processed after their *upstream* neighbors have been processed. Similar dependencies result between processors in which communication is required to pass boundary data *downstream* and whose cost is typically impacted by the slowest communication channel in use. In this work we develop a novel hierarchical wave-front approach that reduces the use of slower communications in the hierarchy but at the cost of additional steps in the parallel computation and higher use of on-chip communications. This tradeoff is explored using a performance model. An implementation using the reverse-acceleration programming model on the petascale Roadrunner system demonstrates a 27% performance improvement at full system-scale on a kernel application. The approach is generally applicable to large-scale multi-core and accelerated systems where a differential in communication performance exists.

1. Introduction

Today's large-scale systems increasingly exhibit differences in their communication capabilities at different levels of the system hierarchy. Highest performing communications, with low latency and high bandwidths, occur between processor-cores that are located on the same chip, and lower performance communications, with higher latency and lower bandwidths, occur between processor-cores located on different compute-nodes. The difference in communication performance is over two-orders-of-magnitude on several of today's systems. A key challenge in the utilization of these systems lies with the efficient use of the available communication channels with differing performance characteristics.

As the number of cores on-chip increases, in line with Moore's law, more complex high-speed communication topologies are being proposed and are appearing in main-stream processors. Examples include the IBM Cell Broadband-Engine [1], and its latest implementation the PowerXCell 8i, which has an on-chip ring network, the Intel Larrabee processor [2] also with an on-chip ring network, and the TILE64 [3] processor with its on-chip mesh network. These networks allow very high speed data transfers between cores on a chip and to shared resources such as controllers to off-chip memories. However the communication performances to other parts of a system degrade significantly and can pose a bottleneck for many applications.

* Corresponding author.
E-mail address: darren.kerbyson@pnl.gov (D.J. Kerbyson).

0167-8191/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2011.02.008

In this work we examine a class of applications that contain wave-front processing. These applications are characterized by a dependency in their processing flow in which grid-points can only be processed after their upstream neighbors. When mapped to a parallel system a local sub-grid can only be processed after boundary data is received from upstream neighbors, and produces boundary data for downstream processors. Blocking of the local sub-grid has been shown to be beneficial in improving the parallel efficiency on large-scale systems [4]. The wave-front processing corresponds to that of a pipeline which takes time to fill and reach full utilization. The slowest communication channel used impacts the performance of the pipeline processing. It has been estimated that a significant number of cycles on the Advanced Simulation and Computing (ASC) machines are used to process applications with wave-front processing [5].

A novel wave-front algorithm, termed the hierarchical wave-front, is developed and implemented in this work. The hierarchical wave-front performs the same processing requirements as in the standard approach and preserves all data dependencies. It uses knowledge of a local processor-core *domain* that typically contains all cores on the same chip. In the hierarchical wave-front the number of slower, inter-domain, communications are reduced but at the expense of increasing the number of parallel computation steps and increased intra-domain communications. A trade-off results between the savings in communication and increased on-chip activities. A performance model is used to quantify this trade-off and show that, for a range in the performance-space consisting of core computation performance, inter-domain communication performance and blocking factor, there is a system-size above which performance improvements will result when using the hierarchical wave-front.

The hybrid petascale Roadrunner system [6] at Los Alamos is used as a test bed to demonstrate the performance improvements that can be obtained in practice. This, a hybrid system containing both AMD Opteron host processors and PowerXCell 8i accelerators, sees a performance improvement of 27% at full-system scale when using the hierarchical wave-front. But performance improvements only occur when using more than 16% of the system. The implementation uses the reverse-acceleration programming model [7] in which each core of the accelerator is a separate MPI rank, and host processors simply support their activity.

The paper is organized as follows. In Section 2 we provide an overview of wave-front processing and describe how the standard implementation is modified to implement the hierarchical wave-front. A performance model is used to compare the performance potential of the hierarchical wave-front in Section 3. An overview of the Roadrunner system is provided in Section 4 illustrating programming models that can be employed and also the performance of its communication hierarchy. A performance comparison of the wave-front implementations on Roadrunner is given in Section 5 that shows that significant improvements are achievable from the hierarchical wave-front.

1.1. Related work

The hierarchical wave-front introduced here is one of few recent works in which specific resources provided by multi-core processors are utilized to optimize scientific application performance. The available parallelism in these processors is being used by most to improve performance, and dedicating cores to specific activities is being explored by many. The closest related application work optimizes the temporal re-use of cache in multi-core processors [8,9], unlike ours which optimizes for the communication hierarchy.

There has been much analysis using wave-front algorithms, the early work using large-scale systems stems back to radiation transport simulations in three-dimensions using the kernel application, Sweep3D [4]. One of the earliest detailed performance analysis and performance modeling of wave-fronts was undertaken in [5] and has subsequently been used in large-scale system procurement as well as in exploring the design space of possible future systems including accelerated systems [10]. The Smith–Waterman algorithm for sequence alignment also includes wave-front processing but on two-dimensional data. Its performance has recently been explored for use on accelerators but at small scales [11].

Recently, there have been several implementations of Sweep3D for the Cell Broadband-Engine including that by IBM [12], and that by Los Alamos [13]. The IBM implementation focused on a single cell processor and required excessive data motion resulting in sub-optimal performance. The Los Alamos implementation followed a distributed memory approach as suggested earlier in [10] and had minimal data movement resulting in 3x higher performance and which has subsequently been shown to scale on Roadrunner [6]. This implementation assigned a static sub-grid to each SPE in the cell processors using familiar MPI style message passing for communications. This work also spawned the Cell-Messaging-Layer (CML) [14], an implementation of the Reverse-acceleration model [7], that provides a lightweight MPI library for the cell. Both the Los Alamos port of Sweep3D to the cell and CML are used in this work.

Further implementations of Sweep3D have also been achieved on GPUs including that on the Nvidia GT200 using CUDA [15]. This demonstrated a speedup of 2.25 over the use of a single contemporary Intel CPU but only at small scale. It will be interesting to see if these results also extend to large-scale systems such as our work here on the large-scale Roadrunner system.

As will be quantified in Section 3, our hierarchical wave-front does not always result in increased performance, but rather results in a complex trade-off between reduced communication and increased on-chip activity. This trade-off is quantified for a large performance-space covering many of the large-scale systems available today and foreseen in the near-future. The performance model is validated and shows high correspondence to actual performance measurements. The use of Roadrunner, though a hybrid system with conventional and accelerator processors, illustrates the potential for using the hierarchical wave-front for any multi-core processing system which exhibit a similar communication hierarchy.

2. Hierarchical wave-front algorithm

In the following we make use of an important concept – that of a processor-core *domain*. Processor-cores within a domain (typically all of the cores on the same chip or socket) are able to pass information between each other at much higher speed than processor-cores that are in different domains (typically other socket or compute-node). The hierarchical wave-front approach directly exploits processor-core-domains by reducing the number of slow, inter-domain communications, but at the expense of increasing the number of parallel computation steps and increased intra-domain communications. This is a complex tradeoff that involves many parameters, some are determined by the performance characteristics of the system, and some that are tunable whose optimum values depend on the system-scale, as will be shown in Section 3.

2.1. Wave-front processing

Wave-front algorithms are characterized by a dependency in the processing order of grid-points within a spatial domain. Each grid-point in a multi-dimensional spatial grid can only be processed when previous grid-points in the direction of processing flow have been processed. Examples are shown in Fig. 1 for 1-dimensonal, 2-dimensional, and 3-dimensional regular spatial grids. In each case, five steps of wavefront propagation are shown. For each step, the cell(s) that can be processed are shown in black, and previously processed cells are shown shaded (for the 1-D and 2-D cases). The direction of the wavefront is from left to right (1-D), from lower-left to upper-right (2-D) and from the nearest upper corner into the page (3-D). The so-called wavefront thus moves across the spatial grid in the direction of travel, entering at one corner point and exiting after passing through all cells.

The direction of wavefront travel may vary from one calculation phase to another. It has been noted that the available parallelism, that is the number of spatial cells that can be processed simultaneously is a function of the dimensionality of the spatial grid minus one. We consider below the use of a 3-dimensional grid that corresponds to that used by Sweep3D.

To compare the standard and hierarchical wave-fronts consider the logical six by six processor-core array shown in Fig. 2. This array is partitioned into two by two domains, each containing nine cores, as indicated by the thick lines. Each processor-core is assigned a sub-grid of size $I_s \times J_s \times K$ of the global grid of size $I \times J \times K$ where $I_s = I/P_x$, $J_s = J/P_y$ and P_x, P_y are the processor core counts in the logical two dimensional array. The sub-grids are processed in blocks of size B k-planes (B layers of the sub-grid in the K dimension) at a time which, as described in [4], increases parallel efficiency. Each k-plane in a block consists of $I_s \times J_s$ grid-points.

In Fig. 2 the wave-front computation travels to the lower-right corner from the upper-left with colors indicating which k-plane, in which block, each core is processing in any step. Note that in this example there are 7 blocks each with 4 k-planes resulting with each core processing exactly 28 k-planes of size $I_s \times J_s$ in both algorithms. But the number of computation steps, and number of inter-domain communication varies.

The standard wave-front algorithm is illustrated in Fig. 2(a) noting that each block in this example contains four k-planes and thus a block takes four k-plane steps to process. Communications occur between processor-cores every four steps, and inter-domain communications occur from step 12 onwards. In this example the standard wave-front requires a total of 68 k-plane steps, of which the first 64 are shown in Fig. 2(a), and 11 inter-domain communication steps.

The hierarchical wave-front algorithm is shown in Fig. 2(b) using the same configuration. In contrast to the standard wave-front, after each k-plane is processed boundary information is communicated to downstream processor-cores if they are within the same domain. Boundary information between domains is communicated only when all cores within a domain have processed the same block of their respective sub-grids. Thus inter-domain communications occur after steps 8, and 16 (and subsequent multiples of eight steps). Individual k-plane steps are used to illustrate this processing flow in Fig. 2(b) up to k-plane step 18. A total of 72 k-plane steps are required by the hierarchical wave-front algorithm (an increase from 68) but only 7 inter-domain communication steps are required (a decrease from 11). Using larger grid-sizes on larger-scale systems increases these effects.

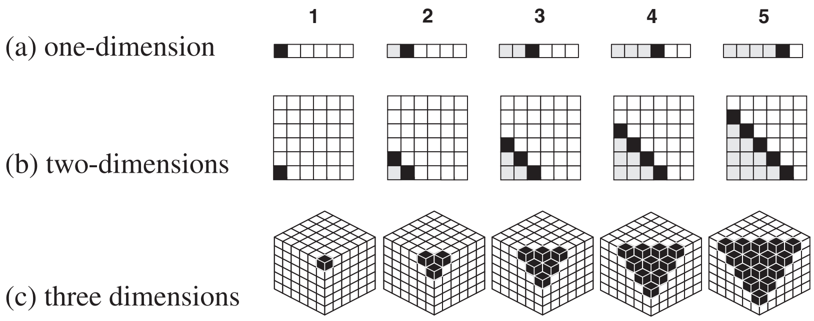
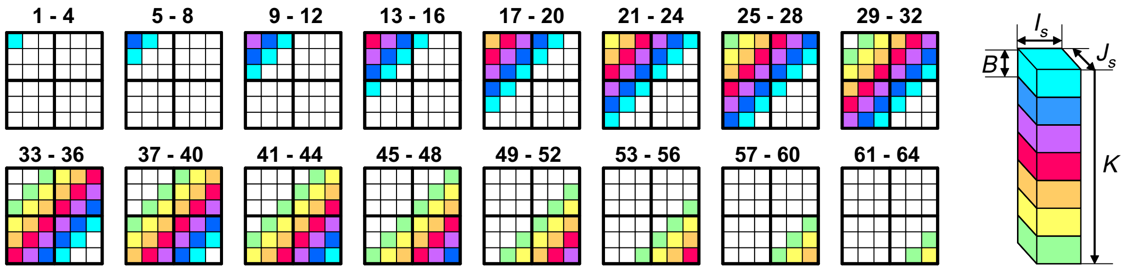
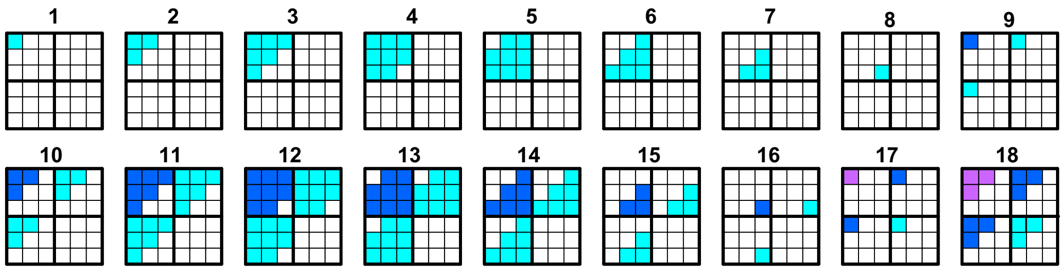


Fig. 1. Example wavefront propagation showing available parallelism.



(a) standard wave-front showing the first 64 k-plane steps (every 4 steps are identical)



(b) hierarchical wave-front showing the first 18 k-plane steps

Fig. 2. Comparison of the standard and hierarchical wave-front algorithms.

2.2. Hierarchical wave-front implementation

An overview of the standard wave-front algorithm is shown in pseudo-code in the top-left of Fig. 3. The main unit of computation is processing a block of the local sub-grid. This is preceded by receiving boundary data from up-stream processors, in this example from the upper and left neighbors, and is followed by sending boundary data to down-stream processors. An example logical 4×4 processor array is also shown in the lower-left of Fig. 3 to illustrate that processor-cores process different blocks at any time (indicated by the different colors), with the cores on the same diagonals processing the same blocks.

The hierarchical wave-front introduces both message aggregation and micro-blocking to the standard algorithm as shown in the upper-right of Fig. 2.

Message aggregation: Only one processor-core within a processor-domain receives boundary data from the up-stream domain in the horizontal dimension and only one core for the vertical dimension. After receiving the boundary, subset are transferred to each other core on the domain edge using high-speed intra-domain transfers. Similarly, one processor-core sends resulting boundary data to the down-stream domain in the horizontal dimensions and one core sends in the vertical dimensions after receiving a subset of the data from each core on the domain edge. For simplicity, the core receiving data from the up-stream domain (and the core sending down-stream) is the same for both dimensions in the lower-right of Fig. 3. A domain of 4×4 processor-cores is assumed in this example. The aggregation reduces pressure on the inter-domain communication sub-system by having only one message sent and received in each dimension, and also results in higher achieved bandwidth on the communication channel due to larger payload sizes. Note that the total amount of data transferred between domains remains the same in both the standard and the aggregation communication schemes.

Micro-blocking: A block is sub-divided into the smallest unit possible – that of a single k-plane (hence forming a *micro-block*) as shown in the upper-right of Fig. 3. Further communications are introduced into the main block processing loop which receive micro-block boundary data from neighboring upstream cores within the domain (if there are any), and also which send micro-block boundary data to neighboring downstream cores (if there are any) as shown by the short dotted arrows in the lower-right of Fig. 3. These additional communication steps are between processor-cores within the same domain using high-speed intra-domain transfers. The micro-blocking results in higher efficiency of the processor-cores within a domain by more rapidly providing work for them to process while at the same time does not require any low-speed inter-domain communications.

3. Potential performance improvement

Prior to implementing the hierarchical wave-front algorithm we employed the use of a performance model. The model enabled us to quantify the potential performance benefits of the new approach and to more fully understand the trade-off between the reduction in the inter-domain communications at the expense of increasing computation as well as

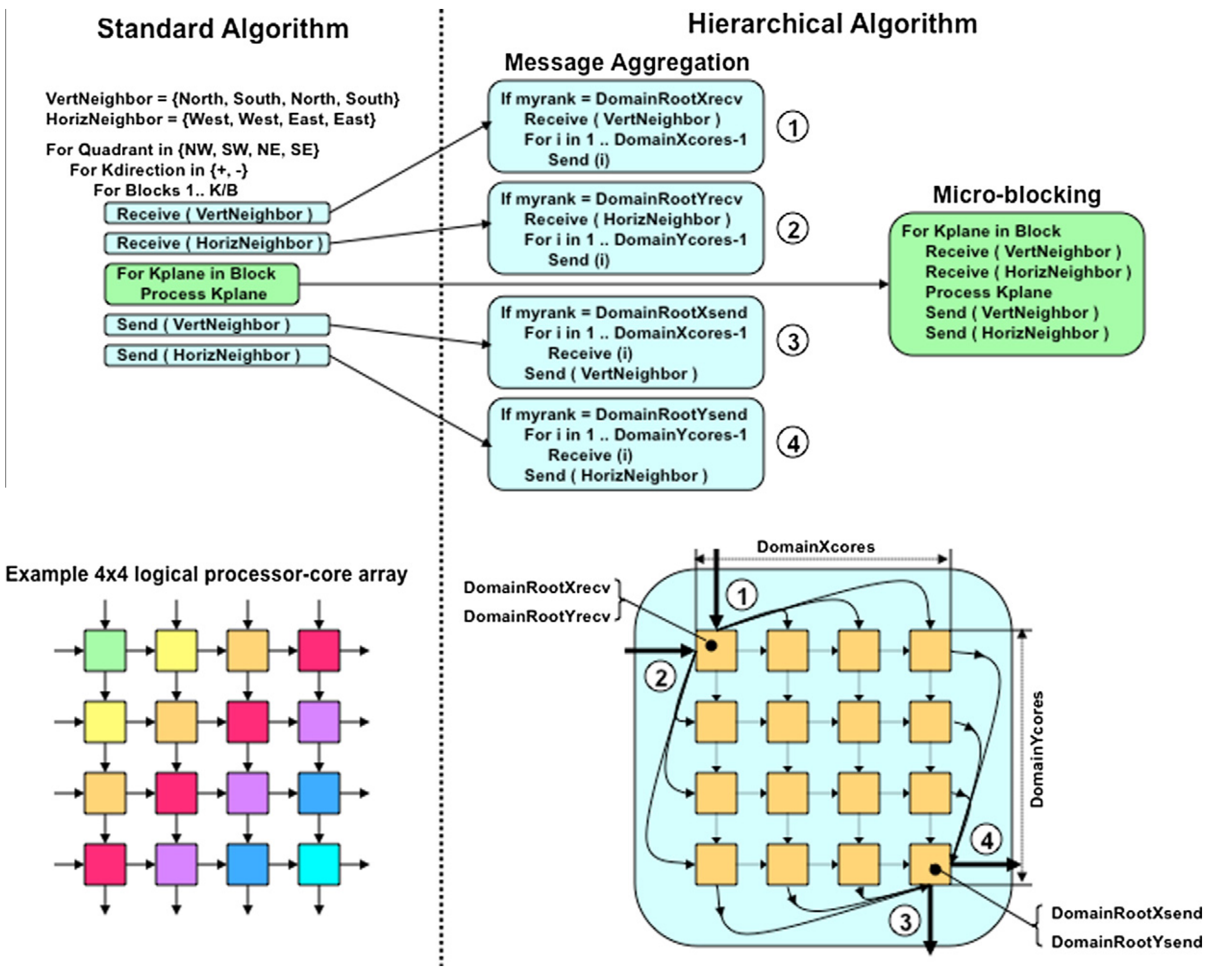


Fig. 3. The standard wave-front algorithm and the main additions required in the hierarchical algorithm. The communication flow between processors in the standard algorithm and in the hierarchical algorithm is also shown.

intra-domain communications. A modified form of the performance model of Sweep3D, as introduced in [5] and subsequently applied to analyzing systems using the Clearspeed CSX600 accelerator [10], was used for this purpose.

3.1. Standard wave-front algorithm

The basic performance model to process a single wave-front [5] is given by:

$$T_{1\text{-}Wavefront} = (P_X + P_Y - 2) \cdot (B \cdot T_{k\text{-}plane} + 2 \cdot T_{msg}) + \frac{K}{B}(B \cdot T_{k\text{-}plane} + 4 \cdot T_{msg}) \quad (1)$$

where the available processor-cores are logically arranged in a two-dimensional array defined by $P_X \times P_Y$. Each computation step takes $B \cdot T_{k\text{-}plane}$ seconds where B is the number of k-planes in a block, and $T_{k\text{-}plane}$ is the time taken to process a single k-plane on a single processor-core. K is the total number of k-planes, and K/B gives the total number of blocks to be processed. The first part of Eq. (1) represents the cost of the wave-front propagating across the processor array, commonly referred to as the pipeline length, while the second part represents cost of processing all blocks locally on a single processor-core. Two boundary communications are required in each step in filling the pipeline, and four (two receives and two sends) are required in addition to the block processing time. The cost for a single communication is T_{msg} and is a function of the block size. T_{msg} is approximated by a two component model in which the first is the message latency (or start-up cost), and the second is the message size divided by the communication channel bandwidth. As we will show later in Section 4, both the message latency and the message bandwidth can significant vary across a system's communication hierarchy.

Eq. (1) represents the case of a single wave-front in which processing originates at one corner of the logical processor array only. In applications such as Sweep3D [4] wave-fronts originate from all corners of a 3-dimensional grid, in a defined order starting with the North-West corner then South-West them North-East then South-East, increasing the number of blocks processed by a factor of eight and also increasing the pipeline length as follows:

$$T_{8\text{-}Wavefront} = (2P_X + 4P_Y - 6) \cdot (B \cdot T_{K\text{-}plane} + 2 \cdot T_{msg}) + 8\frac{K}{B}(B \cdot T_{K-plane} + 4 \cdot T_{msg}) \tag{2}$$

In the following analysis we use the eight wave-front version of the performance model. By examining either Eq. (1) or Eq. (2) it can be seen that the pipeline overhead can be minimized by making B small and hence increasing the relative contribution of the second term to $T_{Wavefront}$. However, this also results with an increased number of communications whose own contribution is minimized by making B large. In actual fact B can be used as a tuning parameter to minimize $T_{Wavefront}$. The optimal value of B generally decreases with the processor scale ($P_X \times P_Y$).

3.2. Hierarchical wave-front algorithm

In order to analyze the hierarchical wave-front algorithm we start with Eq. (2) and consider separately the computation and communication contributions to the overall time. The computation time in the standard algorithm, from Eq. (2), is

$$T_{8\text{-}compute} = \left(2P_X + 4P_Y - 6 + 8\frac{K}{B}\right) \cdot B \cdot T_{K\text{-}plane} \tag{3}$$

The number of computation steps required in the hierarchical algorithm impacts this in two ways: firstly the number of steps to process a single block increases depending on the number of processor-cores P'_X and P'_Y within a domain; and secondly the number of pipeline stages in the logical X and Y processor array dimensions are reduced by P'_X and P'_Y, respectively: for simplicity we assumed that P'_X and P'_Y are factors of P_X and P_Y, respectively

$$T_{8\text{-}compute\text{-}H} = \left(2\frac{P_X}{P'_X} + 4\frac{P_Y}{P'_Y} - 6 + 8\frac{K}{B}\right) \cdot (P'_X + P'_Y - 1)B \cdot T_{K\text{-}plane} \tag{4}$$

The communication time in the standard algorithm, from Eq. (2), is

$$T_{8\text{-}comm} = \left(2(2P_X + 4P_Y - 6) + 4\left(8\frac{K}{B}\right)\right) \cdot T_{msg} \tag{5}$$

To examine the communication time of hierarchical algorithm we assume for simplicity that the time for intra-domain communications is small compared to the inter-domain communications and to the computation time required to process a k-plane. This is true in practice for several large-scale systems including Roadrunner as will be shown in Section 4

$$T_{8\text{-}comm\text{-}H} = \left(2\left(2\frac{P_X}{P'_X} + 4\frac{P_Y}{P'_Y} - 6\right) + 4 \cdot \left(8\frac{K}{B}\right)\right) \cdot T'_{msg} \tag{6}$$

The message time, T'_{msg}, represents the time to send a message which is P'_X (or P'_Y) larger than the message in the standard algorithm. But there is only one message of this size in each step, compared to P'_X (or P'_Y) messages per step in the standard algorithm and thus the amount of traffic per step is the same. For simplicity we assume that $T_{msg} = T'_{msg}$. In addition the difference in latencies, due to different paths in the communication fabric between nodes within a system is assumed small. This is a reasonable assumption for fat-tree networks including Roadrunner's, whose latency between any two nodes varies between 2.1 s and 3.9 s [6], but would need to be more carefully considered for mesh-based systems that can have many more hops between nodes.

The increased computation cost in the hierarchical algorithm over the standard algorithm is:

$$T_{\Delta compute} = T_{8\text{-}compute\text{-}H} - T_{8\text{-}compute} = \left(\left(2\frac{P_X}{P'_X}(P'_Y - 1) + 4\frac{P_Y}{P'_Y}(P'_X - 1)\right) + \left(8\frac{K}{B} - 6\right)(P'_X + P'_Y - 2)\right)B \cdot T_{K\text{-}plane} \tag{7}$$

and the decrease in communication cost in the hierarchical case over the standard case, is given by:

$$T_{\Delta comm} = T_{8\text{-}comm} - T_{8\text{-}comm\text{-}H} = \left(2\left(P_X - \frac{P_X}{P'_X}\right) + 4\left(P_Y - \frac{P_Y}{P'_Y}\right)\right) \cdot 2T_{msg} \tag{8}$$

The hierarchical wave-front algorithm results in a higher performance when the savings in the communication cost are greater than the increased cost of the computation, i.e. when

$$T_{\Delta comm} > T_{\Delta compute}$$

An improvement in performance will not always result from the hierarchical algorithm – the improvement is dependent on the first order effects of: the computation cost to process a single k-plane, $T_{k\text{-}plane}$, the communication time, $T_{msg,}$ block size, B, and also on the system scale ($P_X \times P_Y$) as well as the local processor-core domain size ($P'_X \times P'_Y$).

The potential performance improvements of the hierarchical wave-front approach are shown in Fig. 4. A fixed processor domain of 16 cores ($P'_X = P'_Y = 4$) is considered, while the message time is varied from 5 to 70 µs, and the processor-core count is varied from $P_X = P_Y$=4 (16 cores) to $P_X = P_Y = 512$ (256K-cores). Each processor-core was assumed to process a

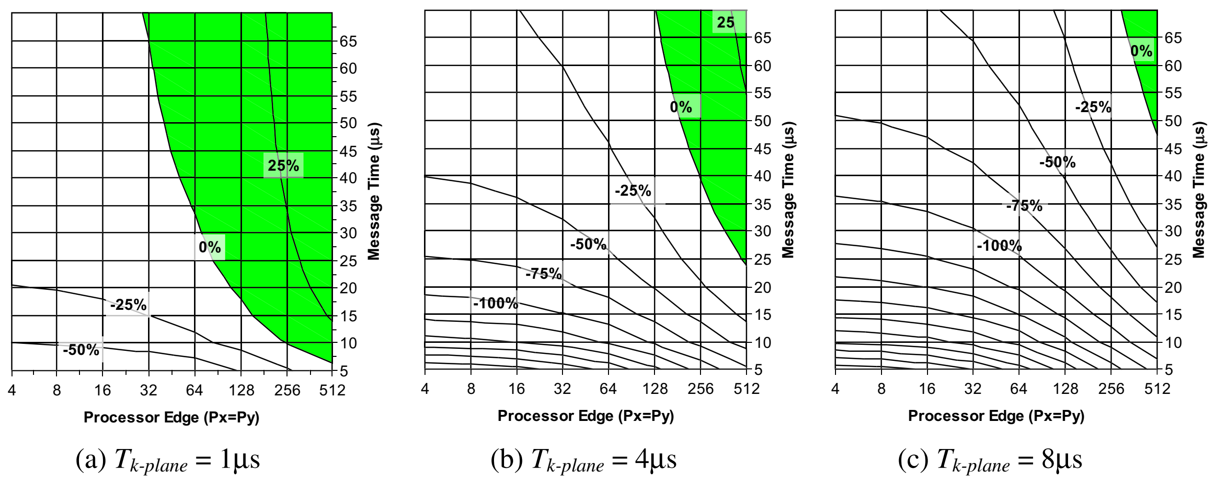


Fig. 4. Potential performance improvement of the hierarchical wave-front for a range of processor-core counts (from 16 to 256K), and range of message times (5 µs to 70 µs).

sub-grid size of 5 × 5 × 400 grid-points with four k-planes per block in all cases. The size of communications is dependent on the sub-grid size and the block size. In the example here, the communication size in the standard algorithm was 960*B*, and in the hierarchical algorithm was 3840*B* for communication between cores across domains, and 240*B* between cores within a domain. Three values for the k-plane computation time are considered – 1 µs, 4 µs, and 8 µs. Note that the values of these parameters are chosen to reflect the costs that are seen in today's systems as well as those that could be expected in near-term future systems.

Performance improvements are indicated by the shaded region in each of Fig. 4(a)–(c). The greatest improvements are seen for the higher processor-core counts where the contribution of the pipeline to the overall execution time is the highest. The hierarchical wave-front is directly aimed at reducing this by lowering the number of inter-domain communications. Performance is lost at lower processor-counts due to the increased computation being greater than the reduction in the savings in the inter-domain communications. Performance is also lost when the communication time is low, as well as when the k-plane computation time is high. The complex interaction between these main performance parameters is clearly shown in Fig. 4.

4. Case study: Roadrunner

We use the Roadrunner system at Los Alamos to demonstrate the performance improvements that are possible from the hierarchical wave-front algorithm. Roadrunner exhibits rich processing and communication resources which vary in their performance characteristics. An overview of Roadrunner along with pertinent performance characteristics are detailed below. A more detailed description of the system architecture can be found in [6].

4.1. Overview of the Roadrunner system

Roadrunner was the first system to achieve a sustained petaflop on the Linpack benchmark. The combination of flexible general-purpose (AMD Opteron) and high-performing special-purpose (IBM PowerXCell 8i [1] – the latest implementation of the Cell Broadband-Engine architecture) processors is the foundation of the Roadrunner system. The goals of the design were to provide high computational performance within acceptable cost and power budgets, and the use of hybrid processor technology was found to be a suitable approach to meet those constraints.

Though Roadrunner contains an equal number of conventional, general-purpose microprocessor cores and special-purpose accelerators the vast majority of the available performance results from the special-purpose accelerators, the PowerXCell 8i processors. These provide over 95% of the peak performance and over 85% of the peak memory bandwidth. The entire system has a peak performance of 1.38 Pflop/s (double precision).

A Roadrunner compute-node is shown in Fig. 5 and consists of three blades. One blade houses the two dual-core Opteron processors, and two further blades each house two PowerXCell 8i processors. The peak performance of a node is 449.6 Gflop/s (double precision). The Opteron processors are clocked at 1.8 GHz with each core able to issue two double-precision floating-point operations per cycle, resulting in a peak of 14.4 Gflop/s across all four cores. The PowerXCell 8i processors are clocked at 3.2 GHz and contain one Power Processing Element (PPE) and eight Synergistic Processing Elements (SPEs). The PPE can issue two double-precision floating-point operations per cycle. Each SPE contains an SIMD processing unit that can issue a total of four double-precision or eight single-precision floating-point operations per cycle. Thus the peak perfor-

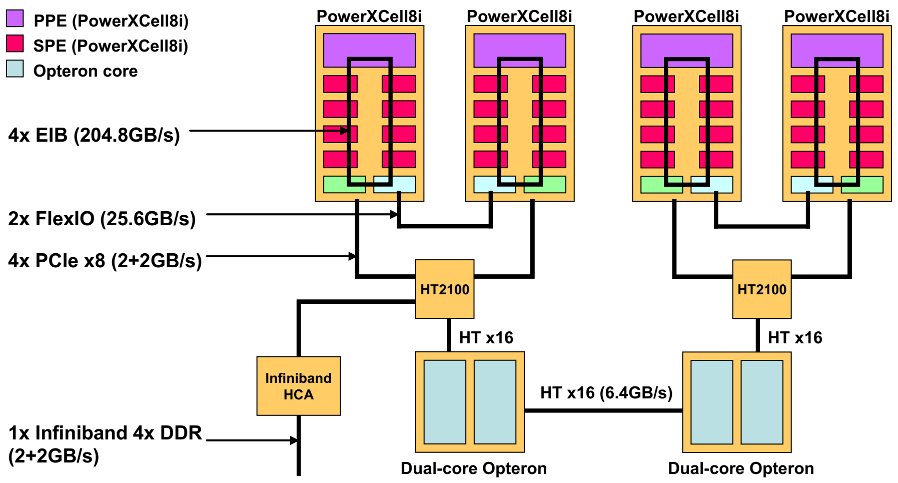


Fig. 5. A Roadrunner compute-node illustrating the peak communication bandwidths between processors both within and between nodes.

mance per PowerXCell 8i is 108.8 double-precision Gflop/s of which 102.4 Gflop/s are from the eight SPEs. The PPE has a traditional cache-based memory hierarchy whereas each SPE can only directly address 256 KB of on-chip (local-store) memory. Main memory, shared with the PPE, can be accessed only via explicit direct memory access (DMA) transfers to or from local store.

The full system consists of 3060 compute-nodes that are arranged into 17 compute units (CUs). The 180 nodes within each CU are interconnected in a full fat-tree topology using a single 288-port InfiniBand 4X DDR switch. CUs are interconnected using a further eight switches organized as a 2:1 reduced fat tree.

4.2. Programming models for Roadrunner

The traditional approach to programming a hybrid systems including Roadrunner is what we term the *accelerator model*. The accelerator model treats the general-purpose cores as main processors and the special-purpose cores as accelerators whose role is to speed up pieces of the application using either data- or task parallel approaches. The enticement of the accelerator model is that unmodified applications can run immediately and performance improvements made by offloading compute-intensive routines to the accelerator can be implemented incrementally. An alternate view is that of the *reverse-acceleration* model [7]. Instead of treating a hybrid system as a cluster of communicating general-purpose cores, each with an attached accelerator for offloading compute-intensive work, one treats a hybrid system as a cluster of communicating, high-speed, special-purpose cores, each with an attached general-purpose core for offloading control-, memory-, or I/O intensive work.

The two programming models are depicted in Fig. 6. In the accelerator model, Fig. 6(a), the general purpose cores (the Opterons in Roadrunner), manage the computation, farming out the compute-intensive work to the special purpose cores (the SPEs in Roadrunner). In the reverse-acceleration model, Fig. 6(b), the special-purpose cores manage the computation farming out control-intensive work to the general-purpose cores and aggregating the results. In the accelerator model, general-purpose cores communicate with other general-purpose cores while the special-purpose cores communicate only with their associated general-purpose cores. In the reverse-acceleration model, special-purpose cores communicate with other special-purpose cores while the general-purpose cores communicate only with their associated special-purpose cores.

The implementation of the standard and Hierarchical wave-front algorithms used the reverse-acceleration model in this work. This was achieved by the use of the lightweight Cell-Messaging-Layer (CML) [14] that implements many MPI functions. In CML tasks are considered to be SPEs with each SPE being given an MPI rank and can communicate to other SPEs in the system. CML manages the communications using the resources provided by the PPEs and the Opterons as needed. The programmer thus focuses on implementing a familiar MPI program on the available SPEs but with the addition of SPE specific optimizations.

4.3. Roadrunner's communication hierarchy

Roadrunner's deep communication hierarchy is also illustrated in Fig. 5. Within a PowerXCell 8i, the SPEs, PPE, and other logic are connected via the Element Interconnect Bus (EIB). The EIB contains four rings (two running clockwise and two counterclockwise) and supports an aggregate peak bandwidth of 204.8 GB/s, although a single transfer cannot exceed 25.6 GB/s [16]. The pair of PowerXCell 8i processors on the same blade are directly connected via a FlexIO interface, providing an aggregate peak bandwidth of 25 GB/s with single transfers limited to 6.25 GB/s. Each PowerXCell 8i blade is connected

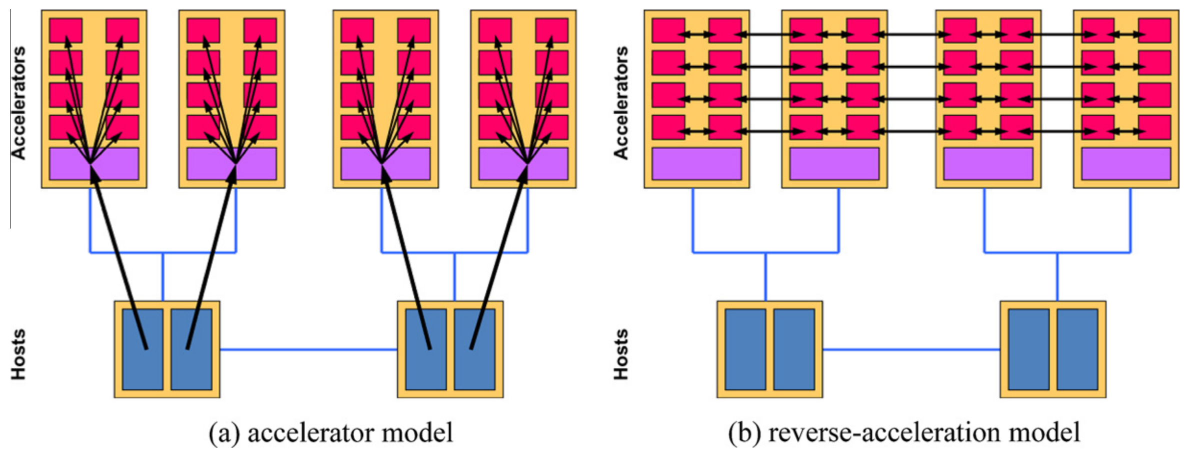


Fig. 6. Comparison of programming models for hybrid systems showing logical connectivity.

to the Opteron blade via two PCI Express (PCIe) ×8 connections as shown in Fig. 5. The PCIe buses from the cell blades are converted to HyperTransport for connection to the Opteron processors using two Broadcom HT2100 I/O controllers. The HT2100 has a single HyperTransport ×16 port and three PCIe ×8 ports. Each PCIe ×8 connection has a peak of 2 GB/s in each direction. A third port on one of the HT2100 connects a Mellanox 4× DDR InfiniBand host channel adapter (HCA). Connectivity between compute-nodes therefore exhibits a peak bandwidth of 2 GB/s in each direction.

Note also that the EIB is shared by the eight SPEs within a single PowerXCell 8i, the FlexIO is shared by the two PowerXCell 8i processors on a single blade, each PCIe is used by only one PowerXCell 8i, and the Infiniband is shared by all processors within a compute-node. After taking this into account the deep communication hierarchy that exists within Roadrunner is even more apparent. There is over two-orders-of-magnitude difference between communication using the EIB and communications using Infiniband.

The actual communication performance that can be realized results from both the peak capabilities of the channels as well as any buffering overheads and underlying system software. On Roadrunner several low-level communication mechanisms are available. MFC I/O [17] for intra-socket communication among SPEs (over the EIB) and between the SPEs and the PPE, the Data Communication and Synchronization Library (DaCS) [18] for communication within a node between PPE and an Opteron, and MPI for inter-node communications.

The performance of each of the low-level communication mechanisms available on Roadrunner is shown in Fig. 7 for transfer sizes between 1-byte and 128K-bytes using a log–log scale. The observed 0-byte latencies and the bandwidth at 128-KB message sizes are summarized in Table 1. Close to peak communication performance is achieved on the EIB and on the FlexIO at 128 KB data transfers with low 0-byte latencies. However, only ~40% of peak bandwidth is achieved on both the PCIe (PPE to Opteron) and Infiniband (Node to Node) communications – larger message sizes are needed to achieve near peak performance.

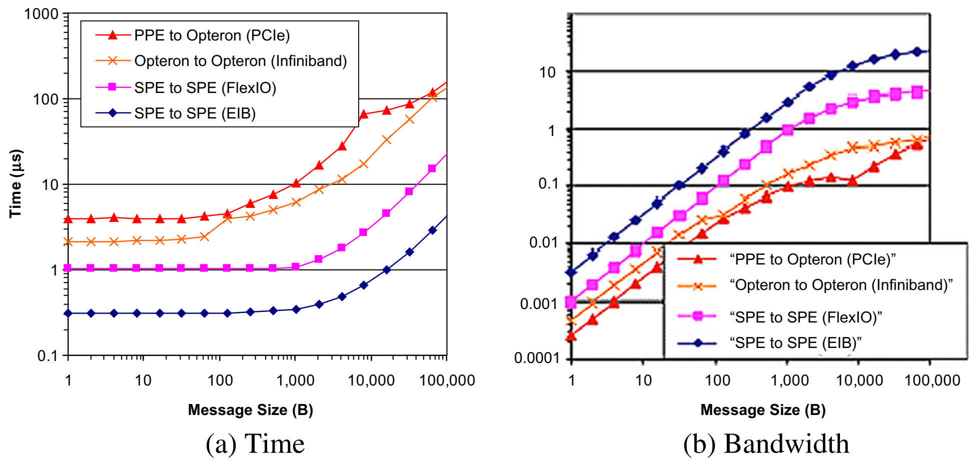


Fig. 7. Measured communication performance of Roadrunner's communication channels.

Table 1
Zero-byte latencies and 128-KB bandwidths of Roadrunner's communication channels.

	Latency (0-B, μs)	Bandwidth (128-KB, GB/s)
SPE–SPE (EIB)	0.3	23.9
SPE–SPE (FlexIO)	0.8	4.5
PPE–Opteron (PCIe)	3.2	0.7
Opteron–Opteron (IB)	2.1	0.8

It is more appropriate to consider the communication cost of the actual transfer sizes used by an application. As described in Section 3, the wave-front algorithms typically require communications of ~1 KB for the standard, and ~4 KB for the Hierarchical algorithm. Bandwidths significantly lower than the peak, on both the PCIe and the Infiniband communication channels, are achieved at these message sizes.

In addition, to communicate from one SPE to another SPE in a different compute-node several communication stages are required: from SPE_1 to PPE_1, from PPE_1 to $Opteron_1$, from $Opteron_1$ to $Opteron_2$, and then from $Opteron_2$ to PPE_2 and PPE_2 to SPE_2 (the source node is denoted by the subscript 1 and the destination by the subscript 2). For a message of size 1 KB this amounts to 30 μs transfer time, and for a 4 KB transfer is 70 μs. Note that there are opportunities to concurrently transfer multiple messages at different stages in this communication flow.

5. Performance comparison and discussion

In order to compare the performance of the standard and hierarchical wave-front algorithms the optimized version of Sweep3D for the PowerXCell 8i was utilized [13]. This version made extensive use of the cell's capabilities to achieve high performance including: explicit management of the local-store using DMAs, cell SIMD intrinsic functions, optimized instruction scheduling, and branch hint instructions. The port to the cell was simplified by the use of the Cell-Messaging-Layer.

The message aggregation and micro-block features of the hierarchical wave-front algorithm, as described in Section 2.2, were added to the cell version of Sweep3D thus taking advantage of the SPE specific optimizations already implemented. The performance of the two versions was measured on the Roadrunner system up to the full system size of 3060 compute-nodes each containing four PowerXCell 8i processors for a total of 97,920 SPEs. In the following analysis a sub-grid of size $5 \times 5 \times 400$ grid-points was assigned to each SPE in a weak-scaling mode, i.e. the global problem scaled in proportion to the number of SPEs used and the problem per SPE remained a constant. A processor-core domain consisted of the 16 SPEs of the two PowerXCell 8i processors on each blade with $P'_X = P'_Y = 4$. The measured grind time on a single SPE was 180 ns for six angles per grid-point resulting with $T_{k\text{-}plane} = 4.5$ μs for a $5 \times 5 \times 400$ sub-grid size. This represents a flop rate of ~5% since 20 flops are required for each angle in each grid point, that includes one division (which takes 9 cycles on the PowerXCell 8i). Though low, this flop rate is comparable to the performance observed on other processors. The communication time, T_{msg}, for a 4 KB message transfer was discussed in Section 4.3.

The performance was measured for both the standard and the hierarchical implementations of Sweep3D for block sizes of $B = 4$, $B = 8$, and $B = 10$ as shown in Fig. 8(a)–(c), respectively. The time for 10 iterations is shown. The increase in time with scale is characteristic of the wave-front algorithm due to the increase in pipeline length with scale and hence increases in communication times that, to some extent, are unavoidable.

It can be seen in Fig. 8 that the hierarchical wave-front is slower than the standard wave-front up to a certain processor-core count. At small scales, the increase in the cost of additional computation steps outweighs the benefit of reduced inter-domain communications (as predicted using the performance model in Section 3). However at larger scales the reverse can be seen, i.e. the reduction in the cost of the communications outweighs the increased computation. The processor-core count at which the hierarchical wave-front achieves higher performance is dependent on the block size – the smaller the block size the earlier the performance improvement occurs.

The relative performance between the standard and the hierarchical wave-fronts is shown in Fig. 9. This is shown in Fig. 9(a) on an equal block bases, as considered for Fig. 8, as well as when considering the best observed performance at each processor scale, over all block sizes, for each of the wave-front types in Fig. 9(b). The thick red-line[1] indicates equal performance between the two implementations, and a value above 0 indicates a higher performance from the hierarchical wave-front. The advantage of the hierarchical wave-front is clear at large-scale especially at the smaller, $B = 4$, block size. Also shown in Fig. 9(b) is the expected performance improvement as given by the performance model for the best block size. It can be seen that there is very good correspondence between the model and the measured performance.

Overall the hierarchical wave-front achieved a higher level of performance on Roadrunner when using more than ~16,000 SPEs. The maximum performance improvement observed was 27% on the full system, and the performance model predicts even higher performance advantages on even larger systems. As a side-note a system with 27% of the peak performance of Roadrunner would itself exhibit a peak of over 370 Teraflops.

[1] For interpretation of color in Fig. 9, the reader is referred to the web version of this article.

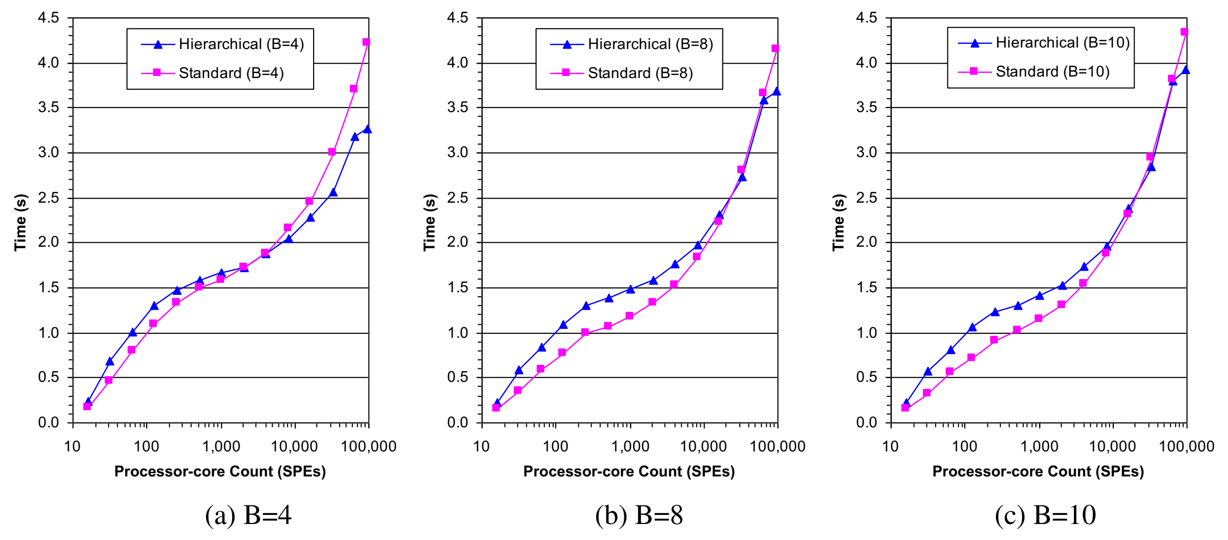


Fig. 8. Measured performance of Sweep3D on Roadrunner for different block sizes.

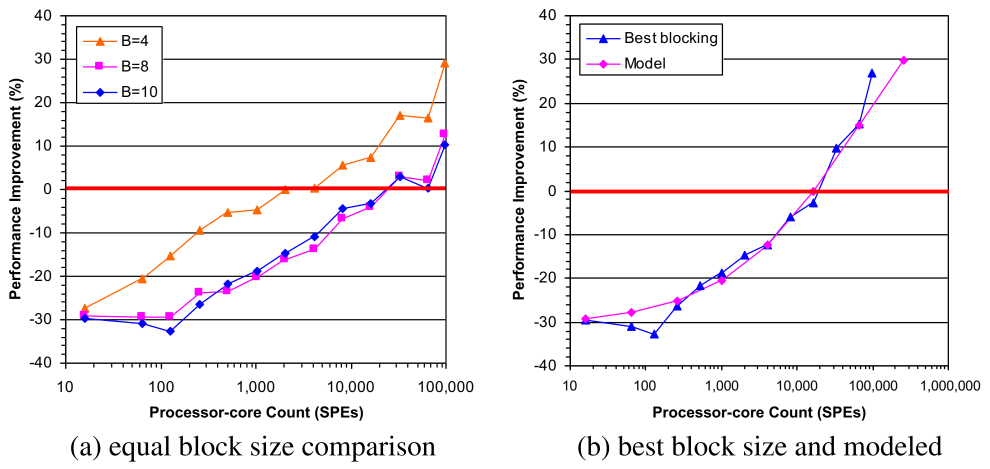


Fig. 9. Performance improvement of the hierarchical vs. standard wave-front implementation.

6. Conclusions

We have shown how significant performance improvements can be achieved for wave-front algorithms on large-scale systems that exhibit large differences in their communication performances. Processor-cores on the same socket are able to communicate at lower latencies, and with higher bandwidths, than cores on different sockets either within the same node or between nodes. We have efficiently exploited this communication hierarchy by developing and implementing a hierarchical wave-front algorithm. In this the number of low-speed communications, between processor-cores in different processor-domains (e.g. sockets or nodes), are reduced but at the expense of increased computation and increased high-speed intra-domain communications. This, a clear trade-off between reduced communication and increased computation, results in higher performance at large-scales when a characteristic of wave-front algorithms, namely the pipeline length, becomes a significant factor.

Using a performance model we initially quantified the potential performance improvements of the hierarchical wave-front by exploring the key parameters of: computation performance, communication performance, and system-scale. Performance improvements were shown to be possible for systems that had large inter-domain communication costs in comparison to the computation performance on a single processor-core and to intra-domain communication costs. The analysis also showed that it is only at large system-scales that performance improvements would occur.

An implementation of the hierarchical wave-front was made starting with an optimized implementation of Sweep3D for the cell. Results from both the standard and hierarchical versions obtained from the Roadrunner system at Los Alamos showed performance improvements when using more than 16K SPEs, with a maximum of 27% improvement observed on the 97,920 SPEs of the full system. These results were in agreement with the initial performance analysis which also indicated larger improvements are possible at even larger scale.

Though the hierarchical wave-front was tested in the hybrid Roadrunner system, the implementation was purely MPI based, using the Cell-Messaging-Layer. Thus, the approach is directly applicable to other large-scale multi-core systems that exhibit similar performance differences in their communication hierarchies. It could also be adapted to hybrid GPU systems but we expect that performance improvements would only result if there were sufficient available parallelism in the sub-grid assigned to each processor-core. The algorithm could also be incorporated into auto-tuning frameworks that consider different optimizations for use at different processor-scales, guiding when and when not to use it.

Acknowledgements

This work was funded in part by the Advanced Simulation and Computing program and the Office of Science of the Department of Energy. It has been authored in part by Battelle Memorial Institute, Pacific Northwest Division, under Contract No. DE-AC05-76RL01830 with the US Department of Energy. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the US Department of Energy under contract DE-AC52-06NA25396.

References

[1] J.A. Kahle, M.N. Day, H.P. Hofstee, C.R. Johns, T.R. Maeurer, D. Shippy, Introduction to the cell multiprocessor, IBM Journal of Research and Development 49 (4/5) (2005) 589–604.
[2] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, P. Hanrahan, Larrabee: a many-core $\times$86 architecture for visual computing, ACM Transactions on Graphics (TOG) 27 (3) (2008) 18:1–18:15.
[3] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, J. Zook, TILE64 processor: A 64-core SoC with mesh interconnect, in: 2008 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, California, 2008. pp. 88–89, 598.
[4] K.R. Koch, R.S. Baker, R.E. Alcouffe, Solution of the first-order form of the 3-D discrete ordinates equation on a massively parallel processor, Transactions of the American Nuclear Society 65 (1992) 198–199.
[5] A. Hoisie, O. Lubeck, H.J. Wasserman, Scalability analysis of multidimensional wavefront algorithms on large-scale SMP clusters, in: 7th Symposium on the Frontiers of Massively Parallel Computation (Frontiers'99), Annapolis, Maryland, 21–25, 1999, pp. 4–15.
[6] K.J. Barker, K. Davis, A. Hoisie, D.J. Kerbyson, M. Lang, S. Pakin, J.C. Sancho, Entering the petaflop era: the architecture and performance of Roadrunner, in: ACM/IEEE SC2008 Conference, Austin, Texas, 2008.
[7] S. Pakin, M. Lang, D.J. Kerbyson, The reverse-acceleration model for programming petascale hybrid systems, IBM Journal of Research and Development 53 (5) (2009) 8.1–8.15.
[8] G. Wellein, G. Hager, T. Zeiser, M. Wittman, H. Fehske, Efficient temporal blocking for stencil computations by multicore-aware parallelization, in: 33rd IEEE International Computer Software and Applications Conference (COMPSAC-2009), Seattle, Washington, 2009, pp. 579–586.
[9] M. Wittman, G. Hager, G. Wellein, Multicore-aware parallel temporal blocking of stencil codes for shared and distributed memory, in: Workshop on Large-Scale Parallel Processing (LSPP), International Parallel and Distributed Processing Symposium (IPDPS), Atlanta, GA, 2010.
[10] D.J. Kerbyson, A. Hoise, A performance analysis of two-level heterogeneous systems on wavefront algorithms, in: E. John, J. Rubio (Eds.), Unique Chips and Systems, Computer Engineering Series, vol. 4, CRC Press, 2007, pp. 259–279.
[11] A.M. Aji, W. Feng, F. Blagojevic, D.S. Nikolopoulos, Cell-SWat: modeling and scheduling wavefront computations on the cell broadband engine, in: 5th International Conference on Computing Frontiers, Ischia, Italy, 2008, pp. 13–22.
[12] F. Petrini, G. Fossum, J. Fernández, A.L. Varbanescu, M. Kistler, M. Perrone, Multicore surprises: lessons learned from optimizing Sweep3D on the cell broadband engine, in: 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007), Long Beach, California, 2007.
[13] O. Lubeck, M. Lang, R. Srinivasan, G. Johnson, Implementation and performance modeling of deterministic particle transport (Sweep3D) on the IBM Cell/B.E, Scientific Programming 17 (2) (2008) 199–208.
[14] S. Pakin, Receiver-initiated message passing over RDMA networks, in: 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, Florida, 2008.
[15] C. Gong, J. Liu, Z. Gong, J. Qin, J. Xie, Optimizing Sweep3D for graphic processing unit, in: Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, LNCS, vol. 6081, Springer-Verlag, 2010, pp. 416–426.
[16] M. Kistler, M. Perrone, F. Petrini, Cell multiprocessor communication network: built for speed, IEEE Micro 26 (3) (2006) 10–23.
[17] International Business Machines, C/C++ Language Extensions for Cell Broadband Architecture, Version 2.5, 2008.
[18] International Business Machines, Data Communication and Synchronization Library for Hybrid – x86: Programmer's Guide and API Reference, Technical document SC33-8408-00. IBM SDK for Multicore Acceleration version 3, release 0, 2007.