Supporting Data-Intensive Applications on Accelerator-Based Distributed Systems

M. Mustafa Rafique (student), Ali R. Butt, and Dimitrios S. Nikolopoulos
Virginia Tech.
E-mail: {mustafa, butta, dsn}@cs.vt.edu

Problem Statement. Multi-core processors can now package several general-purpose processors (GPPs) (e.g. x86, PowerPC) and computational accelerators (e.g. SIMD processors and GPUs), yielding highly power-efficient and cost-efficient designs, with performance exceeding 100 Gflops. These asymmetric accelerator-based processors are rapidly becoming commodity components for high-performance computing, and this trend makes them a viable substitute of less cost-effective alternatives in large-scale clusters. However, the state of knowledge on the use of accelerator-based multi-core processors on large-scale clusters, specially for data-intensive applications, is limited. Current approaches for designing and programming on such clusters are either ad hoc or specific to an installation [1], thus posing several challenges when applied to general setups. First, the effects of alternative workload distributions between GPPs and accelerators are not well understood. Second, accelerators have limited capabilities for managing external system resources, such as communication and I/O devices, thus requiring support from GPPs and special consideration while designing the resource management software. Finally, suitable programming models that adapt to the varying capabilities of the accelerator-type components have not been developed, forcing application writers to micro-manage resources and use platform-specific programming techniques.

Approach. We address these challenges by designing and evaluating alternative asymmetric, accelerator-based cluster configurations for supporting data-intensive applications. We characterize our configurations based on the general-purpose computing and system management capabilities of the accelerators. More specifically, we consider three classes of accelerators: (i) *Self-managed well-provisioned accelerators*, having high compute density, and on-chip capabilities to self-manage I/O and communication. The computational power of the GPP and the amount of available memory is assumed to be sufficient for self-management. Moreover, the tight coupling of accelerators and GPPs enables fast communication between the two types of cores. (ii) *Resource-constrained well-provisioned accelerators*, having high compute density but insufficient on-chip general-purpose computing capability and/or memory for self-managing I/O. These capabilities are provided by an external, dedicated, general-purpose node, which acts as a driver for the accelerators, imposing slower communication paths between the two. (iii) *Resource-constrained shared-driver accelerators*, similar to the previous case, however, they share their drivers with several accelerators, to yield a more cost-efficient design.

We envision four asymmetric clusters configurations for data-intensive applications (Figure 1). *Conf I* consists of self-managed well-provisioned accelerators connected directly to the cluster manager. *Conf II* consists of resource-constrained well-provisioned accelerators. Here, each driver is a powerful resource with large memory and GPP processors dedicated to supporting I/O capabilities to an individual resource-constrained accelerator. The manager distributes the input data to the driver nodes in large chunks, which stream it to the attached accelerators. However, a single manager may not match the data demands of many accelerators simultaneously. *Conf III* addresses this by using a hierarchical setup. Finally, *Conf IV* captures an asymmetric system that may employ a mix of the above configurations as needed.

We develop a MapReduce [3] based programming model that hides the architectural asymmetry while exploiting the computational density of the accelerators. Our design uses a dynamic data streaming approach and uses adaptive resource scheduling that factors in the performance and capabilities of asymmetric components, striving to overlap completely I/O and communication latencies. Our framework implements data transfers and workload scheduling transparently, and adapts the parameters of data streaming and task scheduling dynamically, thereby relieving programmers of some significant programming effort.

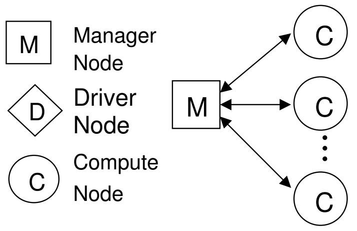
Evaluation. We evaluated our framework under four configurations (Figure 1), using eight Sony PS3s, a manager node, and an 8-node cluster with x86 multi-core processor nodes to serve as drivers. We experimented with a number of typical MapReduce applications [2], to study the effect of the various design alternatives. Figure 2 and 3 show the execution time for *Linear Regression* and *Word Count* benchmarks, respectively, with increasing input size. All four resource configurations scale well with input size for these applications. Our evaluation reveals that our framework exhibits better memory utilization and enables efficient handling of large data sets as compared to a static MapReduce design. Overall we have observed that *Conf IV* offers the best choice: it economizes on the number of drivers, yet provides performance comparable to *Conf II*. We also studied how our framework scales with increasing number of resources, and observed the scaling to be almost linear for all studied benchmarks with an increasing number of compute nodes, up to the point where network bandwidth between the manager, drivers and accelerators is saturated. These results show the promise of our design for use in asymmetric clusters, and we intend to continue exploring and developing the model into a more generic and adaptable framework.

References

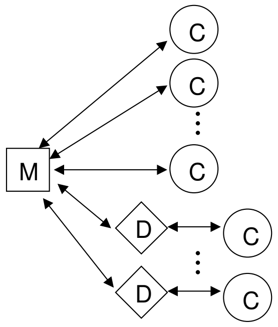
[1] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, and J. C. Sancho. Entering the Petaflop Era: The Architecture and Performance of Roadrunner. In *Proc. SC*, 2008.

[2] M. de Kruijf and K. Sankaralingam. MapReduce for the Cell B.E. Architecture. Technical Report TR1625, Department of Computer Sciences, The University of Wisconsin-Madison, Madison, WI, 2007.

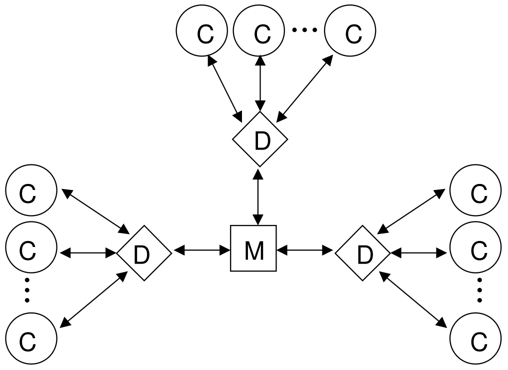
[3] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. USENIX OSDI*, 2004.



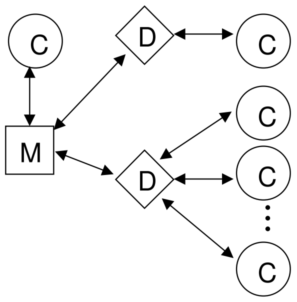
(a) *Conf I:* Self-managed well-provisioned accelerators.



(b) *Conf II:* Resource-constrained well-provisioned accelerators.



(c) *Conf III:* Resource-constrained shared-driver accelerators.



(d) *Conf IV:* Mixed accelerators.

Figure 1: Possible resource configurations in asymmetric clusters.

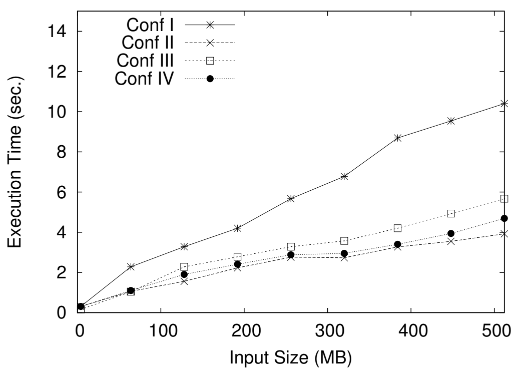


Figure 2: Linear Regression execution time with increasing input size.

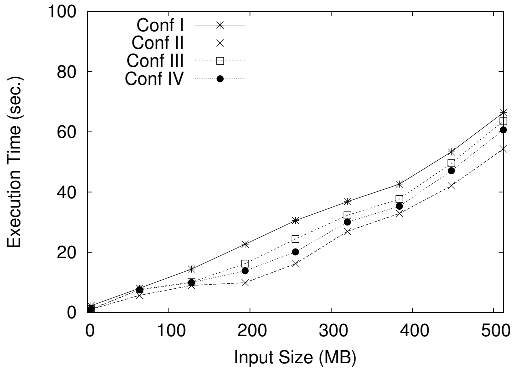


Figure 3: Word Count execution time with increasing input size.