Multi-scale computing over heterogeneous resources

Malte Schwarzkopf † Steven Hand
University of Cambridge Computer Laboratory
{firstname.lastname}@cl.cam.ac.uk
† Student, presenter.

1 Introduction and motivation

Parallel processing of large data sets on clusters of commodity machines—of virtual or physical nature—has in recent years become an active field of research as well as a thriving business. Many companies are running large data centres to support such workloads, and task-parallel frameworks have enabled programmers to harness managed large-scale parallelism while writing mostly sequential code.

At the same time, the increase in individual per-core CPU performance has slowed down significantly and the industry has moved towards parallelism as a way of increasing CPU performance. Non-cache-coherent many-core CPUs are under active research and development, and the average commodity server now contains between four and eight logical CPU cores. On top of this, increasing awareness of data centre energy consumption as an environmental and economic challenge has led to diversification of hardware used in compute and storage clusters [1].

The systems used to run such jobs across many machines, however, essentially treat them as a homogeneous collection: they partition them into many VMs at the hypervisor level or coarse-grainedly break them up into a number of "slots" per machine. Nonetheless, heterogeneity in the data centre is a reality at multiple levels—from hardware and interconnects to task-level heterogeneity within a job [5]—and may even be introduced deliberately [2].

We believe that there has been no appropriate general treatment of resource heterogeneity in compute clusters so far. In this work, we introduce the concept of independent *resource ensembles*, marking a departure from the classic single global master design of task-parallel compute frameworks (§2), and propose a new approach to detecting and characterizing resource heterogeneity within and between ensembles (§3), enabling heterogeneity-aware scheduling decisions. In explicitly supporting heterogeneity-awareness and dynamic extension of a computation across resource ensembles, we introduce the idea of *multi-scale computation*—that is, we can exploit parallelism at many scales, from fine-grained multi-core processing with communication over an on-chip network to coarse-grained clusters of virtual machines (§4).

2 Task-parallelism on resource ensembles

We assume a task-parallel computing framework similar to our previous CIEL system [3]: deterministic task-parallelism introduced explicitly by the programmer, support for dynamic run-time task creation and a distributed object store. Instead of having a single master that coordinates a set of workers, we introduce the concept of *ensembles*: self-organising pools of compute resources that can be nested and can communicate with each other. An atomic compute resource forms an elementary ensemble, and can be aggregated into larger ensembles. Instead of reporting to a global master, resources within an ensemble report to and are managed by its coordinator, and each ensemble coordinator may in turn communicate with other ensemble coordinators.

When a job arrives, its primary computation domain is the ensemble it is first started in. If it turns out that its requirements exceed the capacity of the ensemble, tasks may flow over into other ensembles that have been *peered* with the original one. This could, for example, be another many-core machine in a cluster, or another rack, or a "cloud" service such as Amazon EC2.

3 Describing and detecting heterogeneity

In order to work with heterogeneous resources, we need to be able to characterize their relative performance and capabilities. For this purpose, we employ a combination of *self-description* and *benchmarking*. When a resource

joins an ensemble, it communicates a description of its discrete properties to the ensemble coordinator: e.g. its CPU architecture and model, the number of cores, its network interfaces and their nominal speed, or whether the resource contains any special hardware such as a GPGPU-enabled graphics card.

The ensemble coordinator will then admit the resource, and may ask it to perform a number of *benchmarking tasks*. These are well-defined tasks for which a global knowledge base exists, and which allow the resource to be characterised in terms of its performance completing them. Examples of such tasks may be "compute the 30$^{\text{th}}$ Fibonacci number", "write 100 MB of random output data" or "send data to another resource and measure bandwidth and delay". A similar approach was suggested in the form of a job-specific "calibration phase" by Lee *et al.* [2].

4 Computing at multiple scales

Characterizing resource heterogeneity is useful in itself, and can help to improve scheduling on a heterogeneous cluster. However, clusters are increasingly not just heterogeneous, but also exhibit parallelism at *multiple scales*: intra-chassis parallelism, commonly exploited by multi-threaded applications, and inter-machine parallelism between many machines in a cluster.

In the past, we have experimented with task-parallel programming on Intel's experimental Single-Chip Cloud (SCC) many-core processor using CIEL, and found that a non-cache coherent parallel CPU does indeed present an attractive platform for task-parallel computing, although the assumptions about task granularity and communication channels made in CIEL do not fit it well [4].

We are addressing these issues in our current work: the abstraction of ensembles as sets of resources maps to both intra-chassis parallelism, *viz.* multiple cores, and the set of machines in a cluster. Furthermore, we support different types of communication channels between ensembles. These include on-chip messaging for explicit message-passing architectures like the SCC, shared memory transport for other multi-core architectures, UDP communication on reliable inter-machine links and fully-fledged HTTP RPCs for wide-area channels. Resources will always attempt to use the most high-performance communication channel available between their respective ensembles.

In this setup, ensemble *boundaries* become very important: a computation may traverse such a boundary if it outgrows the capacity of its starting ensemble, as described in §3. However, boundary traversal has several important properties: first, it is only worthwhile if the projected gain in parallelism outweighs the cost incurred by the traversal, and second, a boundary traversal implies a change of fault domain and may also imply a change of administrative authority. Jobs can be constrained in terms of their allowable boundary traversals.

5 Challenges

One outstanding challenge that we are facing in designing a system for multi-scale computing is the one of setting the *task granularity*. Since we are trying to exploit parallelism at multiple scales, the appropriate size of a task may depend on the current scale: when running on a many-core processor with on-chip message passing, the tasks can be much more finely grained than if the communication channel is an HTTP-RPC, as the overheads will dominate in the latter case unless tasks are large. We are looking into different options for dynamic subdivision of tasks as well as support for nested task definitions.

Furthermore, different scales have very different fault-tolerance characteristics, and we are looking into how these can be reconciled to create a system that remains fault-tolerant overall, or which can at least be asked to provide a certain level of per-job fault tolerance guarantee.

6 Further generalisation

Our approach is not limited to the confines of the data centre: today, we find computing resources in many highly diverse environments. It is conceivable to see the set of devices owned by an individual as an ensemble, and to support end-user applications that could operate at multiple scales—for example, a mobile application could harness the computation and storage resources of a desktop machine if written in a task-parallel way and joined to such an ensemble.

References

[1] CHUN, B.-G., IANNACCONE, G., IANNACCONE, G., KATZ, R., LEE, G., AND NICCOLINI, L. An energy case for hybrid datacenters. *ACM SIGOPS Operating Systems Review 44*, 1 (Mar. 2010), 76.

[2] LEE, G., CHUN, B., AND KATZ, R. Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud. In *Proceedings of HotCloud* (2011).

[3] MURRAY, D. G., SCHWARZKOPF, M., SMOWTON, C., SMITH, S., MADHAVAPEDDY, A., AND HAND, S. CIEL: a universal execution engine for distributed data-flow computing. In *Proceedings of NSDI* (2011).

[4] SCHWARZKOPF, M., MURRAY, D., AND HAND, S. Condensing the cloud: running CIEL on many-core. In *Proceedings of EuroSys SFMA* (2011).

[5] ZAHARIA, M., KONWINSKI, A., JOSEPH, A., KATZ, R., AND STOICA, I. Improving MapReduce performance in heterogeneous environments. In *Proceedings of OSDI 2008* (2008), USENIX Association, pp. 29–42.