ETI’s Many-Core Software Architecture for Large-Scale Systems

Executive Summary

ET International, Inc. (ETI) specializes in system software solutions for many-core, high-performance computing systems. Traditional operating systems (OS) are based on the sequential execution model developed in the 1960s and are unable to address new many-core parallel hardware architectures without major overhauls. With the computing industry’s rapid progression toward many-core chipsets and technology, the classical OS and system software are facing unprecedented challenges.

ETI offers an innovative software platform solution designed specifically for many-core systems. Our software works with the classical OS seamlessly and has been deployed on the IBM Cyclops-64 many-core chip, which comprises 160 cores per chip and scales to more than hundreds of chips. This document provides an overview of how ETI’s software system resolves multiple issues facing many-core hardware platforms.

Why Use ETI’s Software System?

How do you harness the power of a >10,000-core system? By using a Linux cluster? Or do you need something else?

With the advent of the many-core revolution, fundamental “technology gaps” must be bridged before applications can fully take advantage of the massive parallelism and high-performance features of multi-core architectures. As noted above, traditional operating systems are based on the 1960’s sequential execution model. These operating systems are at a disadvantage for many-core hardware.

Traditional operating systems have poor scaling capability

Traditional operating systems are intended for single-core processors. Direct porting of existing single- or dual-core-based systems and application software solutions onto a many-core system leads to poor performance that deteriorates further as the number of cores increases. Without major software adaptation, traditional operating systems cannot execute runtime processes efficiently beyond several cores.
The traditional OS model has four major architectural problems:

**Inefficient handling of parallel processing**

Traditional operating systems are architected in ways to optimize resource sharing for multitasking and independent jobs. In high-performance parallel computing, a job is subdivided into a large pool of smaller, related tasks. These tasks interact with each other through a runtime system (RTS). The OS and RTS are unaware of each other, however, and do not share information to optimize performance, but instead often act in opposing ways. For example, a task might be waiting on a resource that is expected to be free in the next few milliseconds; the OS may swap out the task and run something else instead of letting the task continue. The constant swapping of tasks causes unnecessary overhead, and this overhead increases proportionally to the number of tasks running.

The large number of parallel processing tasks becomes very inefficient in traditional operating systems. Each task requires timely access to the underlying hardware at the right time and in the right place, given their mutual dependence.

**Ineffective control structure among cores**

Traditional operating systems have a command-and-control operating structure wherein the OS plays the central administrative and planning role among all threads and cores. This micro-management of local administrative operations ensures task completion but is highly inefficient. To manage all resources in a global manner, the OS uses centralized control and data structures. As the number of cores grows arithmetically, the performance bottleneck grows geometrically, because the central OS is now inundated with request messages.

**Inadequate synchronization and communication among large numbers of parallel tasks**

Traditional operating systems provide very limited synchronization and communication mechanisms among cores. In general, processes that must communicate and synchronize with each other do so through the OS. With the breakthroughs in many-core chip technology including ultra-fast on-chip bandwidths among cores and memory banks, the lack of coordination between cores becomes a real bottleneck. Relying on the traditional OS to coordinate parallel processing will waste a significant amount of communication bandwidth and system resources.
**Abstraction of hardware resources creates unnecessary indirection**

Traditional OS hide the underlying hardware resources from the programmer. Thus, a program would see a four-core environment the same way as a 64-core environment and would react the same way regardless of interfering external programs. While such hardware abstraction enables portability and ease of programming, it limits optimal use of the underlying resources. For example, if a program knew that it was operating on a 32-core environment, it would maximize efficiency by spawning 32 threads so that one runs non-preemptively on each core rather than either spawning more threads and context-switching between them or spawning fewer threads and failing to exploit the available resources.

**ETI’s Solution**

To resolve these shortcomings, ETI has developed an innovative parallel system software solution addressing each traditional OS limitation. ETI’s solution relieves the operating system of micro-management, and has adapted a fine-grained, decentralized software model to unleash the power of many-core parallel computing.

**Runtime Solution Environment**

The motivation behind this decentralized system software model is to provide an on-chip control unit that is adaptable and customizable to the local terrain or cores. This is more efficient and flexible than a conventional, centralized control unit. In our implementation, ETI’s system software relieves the operating system of core-to-core coordination and local resource management. Instead, ETI’s FIRST (Flexible Integrated Runtime System Technology), an execution environment with an embedded micro-OS kernel, manages parallel operations on the chip.

FIRST provides a runtime execution environment that features:

1) fine-grained task scheduling among cores,
2) effective coordination of communication and synchronization among parallel tasks, and
3) dynamic load balancing across the entire system.
In addition to creating a co-processing, layered operating system structure, another distinguishing feature of ETI’s parallel platform is its creation of a fine-grained, dataflow-like, hierarchical thread platform - a clear breakthrough from traditional coarse-grained multithreading.

**Fine-grained parallel system**

The improvement comes in the form of prioritizing the execution of threads based on their “readiness” to be executed. Instead of using sequential program counters as in a traditional OS, ETI’s fine-grained parallel software system executes parallel tasks depending on the “readiness” of the input arguments and available resources.

As an example, in a sequential, counter-based system, similar to cooking a dish, the recipe is broken down into a series of sequential steps such as 1) measuring the ingredient amounts, 2) mixing ingredient A with ingredient B, and 3) adding white wine to the mixture, etc..

In a dataflow execution environment, sequential steps/counters are irrelevant. Instead, inputs are crucial. This is similar to a restaurant’s kitchen with a team of chefs and helpers. Each person works at his or her workstation, and when ingredients enter a workspace, the person stationed begins mixing or chopping these ingredients until done. The output then passes to another person as input. Thus, each person is performing a minute operation independent of the sequential procedure and is dependent only on sufficient inputs. This whole process is analogous to a dataflow environment, wherein each atomic operation is a fine-grained task.

By continuously subdividing tasks to finer-grained tasks and assigning them across many cores as non-preemptible atomic operations, ETI’s FIRST technology has provided a foundation for resolving the scalability issue facing traditional operating systems. The granularity of the tasks is dictated by the hardware environment in such a way as to minimize runtime system overhead (i.e. creating tasks, determining the next non-preemptive task to fire, etc.), while fully exploiting the parallel concurrency of the application.
ETI Software vs. Traditional Software

Why can’t we scale?

<table>
<thead>
<tr>
<th>Issues</th>
<th>Traditional OS</th>
<th>ETI System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Tasks Performance</td>
<td>Parallel tasks as Independent processes</td>
<td>Decentralize with fine-grain atomic based model for parallel tasks</td>
</tr>
<tr>
<td>Core and Task Control</td>
<td>OS Bottleneck: acting as central scheduler</td>
<td>RTS Schedules tasks with awareness of task inter-relatedness</td>
</tr>
<tr>
<td>Cores Synchronization</td>
<td>Inefficient with low bandwidth communicate among cores</td>
<td>Micro-OS kernel for direct on-chip communication and synchronization</td>
</tr>
</tbody>
</table>

Overview of ETI Software Platform

ETI understands both the benefits and the bottlenecks of the traditional OS, and our product is tailor-made for many-core architectures. Two decades of research have led us to develop a dataflow-like software platform for many-core computing. In developing the IBM Cyclops-64 (C64) supercomputer, ETI has been the sole software vendor to design and implement a software infrastructure to unleash the full power of the many-core C64 system.

Our latest version of the system software is designed to work on over one million cores and is currently running on 8000+ cores and over 50 C64 blades. This highly scalable system software includes the following design infrastructure:

- **FIRST (Flexible Integrated Runtime System Technology)** - Delivers a parallel runtime execution model that manages hardware resources efficiently and exposes them to user applications;
- **Control and monitoring software** – Boots the system, ensures safe operating conditions and detects hardware anomalies;
- **Resource manager** – Schedules jobs across the entire system under a multi-user environment;
- **Software development toolkit (SDK)** – Provides a Linux-based environment with which to compile, debug and simulate the many-core chip platforms.

---

1 For more technical information, please see our white paper on Technical Briefing on ETI Software System

2 A blade consists of a C64 chip, a memory unit, an FPGA, and Ethernet. See C64 hardware whitepaper