CCIT REPORT #719 February 2009

Many-Core vs. Many-Thread Machines: Stay Away From the Valley

Zvika Guz[1], Evgeny Bolotin[2], Idit Keidar[1], Avinoam Kolodny[1], Avi Mendelson[3], and Uri C. Weiser[1]

[1]*{zguz@tx, idish@ee, kolodny@ee, uri.weiser@ee}.technion.ac.il , EE Department, Technion-IIT, Israel*
[2]*evgeny.bolotin@intel.com, Intel Corporation, Haifa, Israel*
[3]*avim@microsoft.com, Microsoft Corporation, Haifa, Israel*

Abstract—We study the tradeoffs between Many-Core machines like Intel's Larrabee and Many-Thread machines like Nvidia and AMD GPGPUs. We define a unified model describing a superposition of the two architectures, and use it to identify operation zones for which each machine is more suitable. Moreover, we identify an intermediate zone in which both machines deliver inferior performance. We study the shape of this "performance valley" and provide insights on how it can be avoided.

1 Introduction

As chip multiprocessors are rapidly taking over the computing world, we see the evolution of such chips progressing along two separate paths. At one end of the spectrum, general purpose uniprocessors have evolved into dual- and quad-cores, and are set to continue this trajectory to dozens of cores on-chip. Such machines follow the legacy of single cores in using caches to mask the latency of memory access and reduce out of die bandwidth, and typically dedicate a significant portion of the chip to caches. We call these *Many-Core (MC) machines*. Intel's Larrabee [8] is a prominent example of this approach. At the same time, processor engines that can run numerous simple threads concurrently, which were traditionally used for graphics and media applications, are now evolving to allow general-purpose usage (a trend called GPGPU [10]). The latter usually do not employ caches, and instead use thread level parallelism to mask memory latency, by running other threads when some are stalled, waiting for memory. We refer to these as *Many-Thread (MT) machines*. Examples of such machines are the current GPGPUs of Nvidia and AMD.

To date, the tradeoffs (and even the boundaries) between these approaches are not well formalized. Such formalization and understandings are, nevertheless, essential for processor architects, who need insights on how to improve their machine's performance on a broad range of workloads. In this paper, we take a step towards understanding the tradeoffs between MC and MT machines, and the do-