Improving LUC in Cloud Environments

Kirk W. Cameron and Ali R. Butt
Virginia Tech
{Email: cameron@cs.vt.edu}

Summary

In current work[1], we are exploring how to adapt and design advanced operating systems to maximize performance in power scalable environments. For example, we have observed significant performance slowdowns when CPU power scaling is combined with parallel applications that frequently access I/O. Since power scaling will be essential to future cloud systems to balance infrastructure restrictions on power with performance needs, we must identify the root causes of these types of slowdowns. Our preliminary work shows that identifying root causes requires a combination of exhaustive experimentation with deep knowledge of (and access to) the underlying hardware and software. While our current work has been limited to bare-metal Linux platforms, we have determined slowdowns are highly sensitive to increases in system scalability and variability. Thus, we require experimental platforms of scale that enable: 1) exploration of slowdowns with and without virtualization; 2) exploration of slowdowns in the context of software and hardware heterogeneity; 3) the ability to install and analyze experimental operating and runtime systems; and 4) the ability to measure power consumption and performance. Should the Chameleon and CloudLab platforms provide these capabilities, we plan to make use of them to create new systems that Limit Unintended Consequences (LUC) and maximize performance in power scalable environments.

Preliminary Work

Future cloud systems will be enormous, complex and heterogeneous. It is widely acknowledged that designing efficient systems at scale is a grand challenge for the community [1]. To address inefficiencies, the components of these complex systems perform a growing set of tasks themselves. For example, processors [5], memory [3], and disks [6, 7] are capable of independently managing their power consumption. Such devices attempt to balance user performance demands with the energy constraints of the physical infrastructure.

The combination of device independence with unprecedented degrees of parallelism in software and hardware at system scale leads to ever more complex interactions among operating systems, virtualization, parallel and distributed applications and services, and hardware. Thus, identifying the optimal performance configuration of applications and services on clouds at runtime is exceedingly difficult because the variance among data points may exceed the best average performance operating point.

Figure 1 provides an example of this phenomenon. The figure plots the performance for 64 threads of a parallel transaction workload at various power/frequency settings from 1.6 GHz to 3.1 GHz. Performance is the ratio of speedup versus the slowest frequency and the performance gets worse with higher frequencies. This slowdown is difficult to observe in practice because the standard deviation (shown as the gray/blue area surrounding the line plot) often exceeds the performance difference between two data points.

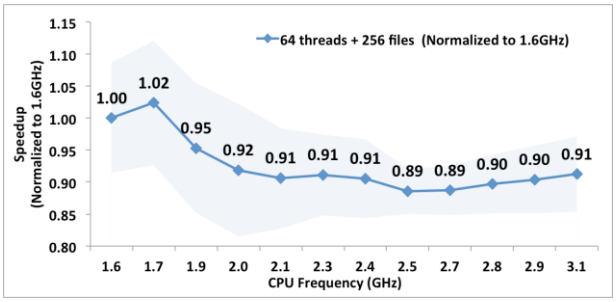

Figure 1 Filebench varmail speedup ratios normalized to the lowest available CPU frequency (1.6 GHz) – higher is better. The grey/blue area shows one standard deviation from the mean. *Nehalem (HDD)* System: Dell T3500 using a W3550 3.00 GHz quad-core with 6 GB of DDR3 RAM and a 250 GB 7200 rpm hard drive.

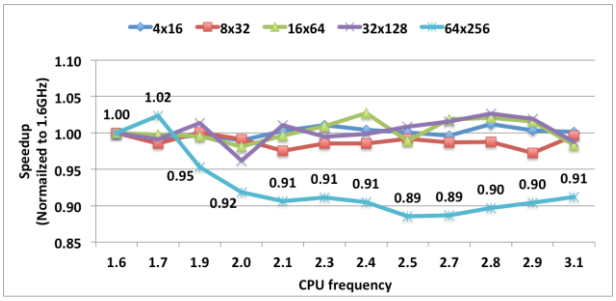

Figure 2 Filebench varmail speedup ratios normalized to the lowest available CPU frequency (1.6 GHz) – higher is better. Each line shows speedup for (number of threads) x (number of files). For 64 threads, performance drops significantly at higher frequencies.

For example, Figure 1 shows the 2.0 GHz frequency has a standard deviation of 12% while 1.7GHz on average performs 10% better. Thus, taking a single measured sample at the low performance range of 1.7 GHz and a single sample at the high performance range of 2.0 GHz results in the flawed conclusion that 2.0 GHz should be selected for better performance. Therefore, a runtime system attempting to adapt processor power/frequency settings [4] and optimize based on sampled runtime performance information could be ineffective and result in significant performance loss.

[1] NSF CSR:Small: Exploiting slowdowns for speedup in power-scalable HPC systems. PI: K. W. Cameron, co-PI: Ali Butt, $500,000, 8/14--7/17.

Additionally, the same code may perform differently at scale. Figure 2 shows the same example from our transaction workload for varying the number of threads from 4 to 64. The slowdown at higher frequencies does not occur until 64 threads where variance between runs increases. Beyond 64 up to 256 threads (not shown) the performance is similar to the 4→32 thread cases.

Figures 1 and 2 show the assumption that increasing power/frequency improves (or at least doesn't hurt) performance is flawed. Without a deeper understanding of the causes of these types of slowdowns and isolation of the root causes, runtime systems cannot avoid these types of slowdowns altogether. A runtime system using sampling to schedule threads runs the additional risk of falling victim to the previously discussed variance issue.

Unfortunately, at the same time complexity threatens system efficiency, performance has never been more critical to the worldwide economy. According to Amazon's Greg Linden and Google's Marissa Meyer[2]: "Amazon found every 100ms of latency cost them 1% in sales. Google found an extra .5 seconds in search page generation time dropped traffic by 20%." High frequency trading also relies on data centers where power management is considered critical yet a few milliseconds of latency loss can result in millions of lost profits[3].

To address the *Unintended Consequences* that are increasing in frequency with the growth in system complexity, we are building runtime systems that identify the causes of performance loss in power scalable systems. For the slowdowns we've isolated so far, causes include 1) additional false sharing in the OS that occurs when processor performance/power is increased; and 2) scenarios where asynchronous I/O is not enabled and hurts performance significantly. Our work [2] shows that our optimized designs that address these two causes result in performance gains from 10-60% depending on I/O workload and system characteristics. For future work, we have also identified a number of other slowdown scenarios that have as yet undetermined root causes.

Our Experimental Cloud Needs

The exhaustive testing necessary for 95% confidence in the results shown in Figures 1 and 2 requires thousands of experiments and takes weeks on our 12-16 node cluster. This is primarily due to the high variance in the measurements that increase with scalability, complexity, and heterogeneity. This requires 40-50 runs per data point for the desired confidence. Currently, we have limited testing primarily to the Linux ext3 and ext4 filesystems since this reduces the problem space. We have also completely ignored the effects of virtualization since 1) the additional abstraction obfuscates isolating performance slowdowns; and 2) the additional software increases the problem space significantly. Currently experiments varying scale and heterogeneity are limited to our testbed, but our results (See Figures 1 and 2) indicate sensitivity to these parameters that must be explored further. We need access to environments (e.g. Chameleon and CloudLab) that allow us to migrate our studies and solutions from bare-metal OS's to virtualized systems. This will enable us to continue to improve the efficiency of future highly scalable and heterogeneous cloud systems and services.

References

1. "Nitrd lsn workshop report on complex engineered networks," 2012; **https://www.nitrd.gov/Publications/PublicationDetail.aspx?pubid=52**.
2. Chang, H., B. Li, M. Grove and K. Cameron, "How processor speedups can slow down i/o performance," *Proceedings of 22nd IEEE International Symposium on Modeling , Analysis, and Simulation of Computer and Telecommunication System (MASCOTS'14)*, IEEE, 2014.
3. David, H., C. Fallin, E. Gorbatov, U.R. Hanebutte and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," *Book Memory power management via dynamic voltage/frequency scaling*, Series Memory power management via dynamic voltage/frequency scaling, ed., Editor ed.^eds., ACM, 2011, pp. 31-40.
4. Ge, R., X. Feng and K.W. Cameron, "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters," *Proceedings of ACM/IEEE International Conferences for High Performance Computing, Networking, Storage, and Analysis (SC05)*, IEEE Computer Society, 2005, pp. 34.
5. Ranganathan, P., P. Leech, D. Irwin and J. Chase, "Ensemble-level power management for dense blade servers," *Proceedings of ACM SIGARCH Computer Architecture News*, IEEE Computer Society, 2006, pp. 66-77.
6. Sehgal, P., V. Tarasov and E. Zadok, "Evaluating performance and energy in file system server workloads," *Proceedings of FAST*, 2010, pp. 253-266.
7. Verma, A., R. Koller, L. Useche and R. Rangaswami, "Srcmap: Energy proportional storage using dynamic consolidation," *Proceedings of FAST*, 2010, pp. 267-280.

[2] http://highscalability.com/latency-everywhere-and-it-costs-you-sales-how-crush-it
[3] http://dealbook.nytimes.com/2014/07/07/no-need-to-demonize-high-frequency-trading/