Estimating System Activity Vectors
Extended Abstract

Alexandros Lagos, Jan H. Schönherr, Jan Richling

Communication and Operating Systems Group
Technische Universität Berlin, Germany
{alexandros.lagos,schnhrr,jan.richling}@tu-berlin.de

The recent trend towards cloud computing allows entities with heavy computing requirements to fulfill those without the need of owning expensive system infrastructures themselves. The ongoing migration of such applications into the cloud leads to situations where multiple parallel applications have to be executed on possibly diverse hardware configurations by the cloud provider. In such environments two aspects are important: performance and energy efficiency.

While each application requires certain resources to perform its work, the total amount of available resources is limited and in modern architectures resources are also segmented. Resources are accessed either synchronously or asynchronously. We are specifically interested in the first case, as stalls due to concurrent accesses cannot be masked by executing something else. Such resources are, for example, not only memory/cache bandwidths or memory/cache sizes, but also execution units or the instruction fetch bandwidth. With multiple tasks competing for resources, resource contention can become a limiting factor reducing performance and, thus, energy efficiency. Therefore, it is paramount to decrease such resource contentions by performing a targeted selection of tasks that are to run concurrently on the same system. And then within the system, the mapping of tasks to CPUs can affect the performance dramatically due to segmented resources. The reduction or even avoidance of bottlenecks can lead directly to increased performance, and thus better energy efficiency and cost savings. The problem is to realize this targeted selection efficiently, without having to actually evaluate all possible task combinations on all available hardware configurations. Such an attempt can be cost and time prohibitive, especially in cloud environments.

We tackle this challenge by creating a model of the resource state of a system. We aim to predict the behavior of executing a combination of tasks without having to actually run those tasks in parallel. Specifically, the model describes the total effects on system resources based on the effects of individual tasks. The accuracy of the model will be determined by comparing the estimated resource utilization against the actual utilization observed during execution.

To capture the resource state of the system, we propose *system activity vectors* (SAVs), a variant of *task activity vectors* [2]. They contain for every resource a percentage value that describes how much of that resource is currently used. For segmented resources, there is a value for each segment. Please note, that due to our model it is not necessary to actually measure SAVs; measurements are only used for evaluation purposes. The resource requirements of tasks are captured in *task activity vectors* (TAVs). Compared to SAVs, TAVs usually represent segmented resources with only one value, as a single task is architecturally restricted to a single segment. The values within a TAV are assumed to be free from external influences. That means on the one hand, that the TAV was measured in isolation, and on the other hand, that tasks are independent from each other.

Our model is based on the premise that every task reaches a resource bottleneck as otherwise its performance would be unlimited. Hence, the consumption of other resources by a task is limited by the bottleneck resource. In fact, the consumption of non-bottleneck resources of a task is in certain cases proportional to the consumption of its bottleneck resource. We further assume, that accesses to bottleneck resources by multiple tasks are resolved by the system weight-proportionally, e. g., when one task would need 50% of a resource and another task 75%, they will receive 40% and 60% of the resource, respectively. Experimental testing has confirmed that this is in many cases a valid assumption. Finally, we cover time/space trade-offs (e. g., increased consumption of memory bandwidth due to reduced amount of cache) by augmenting our TAVs accordingly. By adding individual TAVs to an, in the beginning, empty SAV, we estimate the theoretical resource utilization. When a bottleneck is reached due to the combination of tasks, this can then be seen in the SAV. In that case, we readjust the TAVs responsible for the bottleneck. This may have to be repeated because of the time/space trade-offs. In the end, the SAV contains a numerical representation of the total system resource utilization that would occur when the individually measured tasks were executed in parallel.

Task activity vectors themselves are built by collecting data for a single running task via hardware performance counters provided by modern processors. However, performance counters alone are not yet sufficient to gather all necessary data. Additionally, time/space trade-offs are task specific. So, in order to derive the missing information, we employ well-defined background loads and observe the effects via performance counters.

Presently, we are verifying our model and evaluate it using tasks with constant TAVs. However, in practical cases, tasks contain several phases with varying resource needs. Such tasks cannot be represented accurately by a single TAV. Furthermore, tasks with dependencies cannot be isolated and analyzed directly. To address these issues, work is ongoing towards automatically detecting individual phases within a task and creating appropriate TAVs that enable a far more fine-grained task analysis. Additionally, as previous research has shown, it is possible to gain the energy requirements of the system through performance metrics [1]. The model can then be used to optimize energy efficiency directly.

We think that by modeling the resource state through SAVs, it will be possible to perform task selections for cloud environments that improve energy-efficiency and performance without the added overhead and complexity of testing all possible combinations of tasks. Especially in cases of running multiple instances of the same task, the task needs only to be evaluated for each different hardware configuration that it is actually executed on. Furthermore, in cloud scenarios where a large selection of tasks has to be executed, this approach allows to put those programs together on machines where the resource requirements match best. This can directly lead to increased cost savings while maintaining a high level of service quality to clients.

References

1. Canturk Isci and Margaret Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, pages 93–, Washington, DC, USA, 2003. IEEE Computer Society.
2. Andreas Merkel and Frank Bellosa. Task activity vectors: A new metric for temperature-aware scheduling. In *Third ACM SIGOPS EuroSys Conference*, Glasgow, Scotland, March 31–April 4 2008.