Spatial Resource Allocation on Manycore Platforms Using Predictive Performance Models

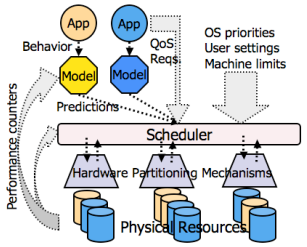
Henry Cook, Sarah Bird, Krste Asanovic and Dave Patterson

PROBLEM STATEMENT

On manycore platforms the OS must now manage both spatial and temporal resource allocation of shared physical resources. At the same time, the growing prevalence of mobile devices has made power and energy first class citizens. How can we get efficient execution on a diversity of platforms with applications that have different resource usage patterns?

The combinatorial scheduling problem worsens if all possible allocations of resources have to be tested at runtime. Instead, we propose to predict the effects changing an allocation using performance models.

DESIGN OVERVIEW



We collect application behaviour and use it to create predictive models. We use the models as input to the scheduler to make decisions. Model accuracy is enhanced by the performance isolation provided by a set of hardware partitioning mechanisms.

PARTITIONING MECHANISMS

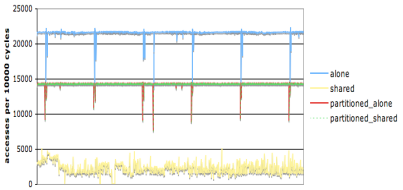
Core Partitioning:
Easily partitioned by assigning threads to cores in a partition. Application chooses which threads run on which cores.

Cache Capacity Partitioning (for shared caches):
Caches can be partitioned by ways or banks. For manycore chips we can use bank-based, allowing an application to be allocated more local banks.

Bandwidth Partitioning:
Using Globally Synchronous Frames (Lee et al. ISCA 2008) we can guarantee minimum bandwidth (Packets/Frame) and bound maximum delay, while also providing differentiated services.

PARTITIONING EVALUATION

This graph shows that our GSF mechanism succeeds in partitioning BW.



MODEL FORMULATION

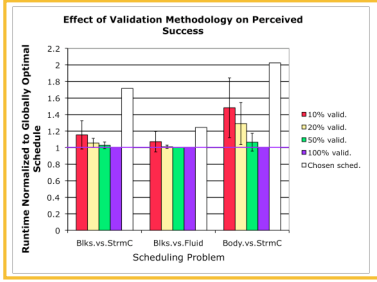
- Create models from performance data sample
  - Input: performance and activity metrics for a sample of possible allocations
  - Output: predicted perf. for untested allocations
- Explore different response surface model types
  - Linear, Quadratic, KCCA, GPRS
- Use models to predict the perf. of possible allocations

METHODOLOGY

- We use RAMP Gold with hardware partitioning
  - Using PARSEC and Synthetic Benchmarks
  - Running Tessellation (ROS)

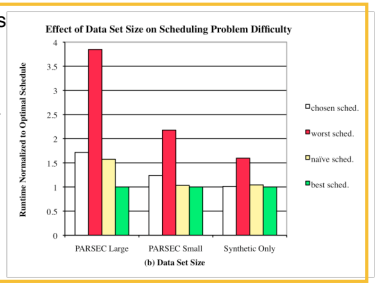
- Collect performance data to create the models.
- Collect performance data for all possible allocations to validate models and decisions

SAMPLING SENSITIVITY ANALYSIS



This figure illustrates the effect of validation set size on perceived success. Our chosen allocation appears to be very close to optimal when only sampling a small number of points.

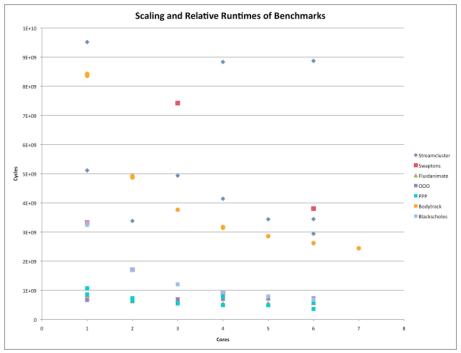
This figure illustrates the effect of the benchmark input data set size on the scheduling problem. For small data sets the best and worst case are so close that models is too expensive. This is not true of the large size.



CONCLUSIONS

- Scheduling using predictive performance models shows a lot of promise.
  - Quadratic model is within 3% of optimal
  - Time-multiplexing is on average 2x of optimal
  - Dividing the machine in half is on average 1.5x of opt.
- It's important to evaluate the approach on a system with full size benchmarks and testing all the allocations

BENCHMARK SCALING



This graph shows the performance of our bmks for different allocations. Points with the same core count but different performance have different cache or bandwidth allocations.

MAKING SCHEDULING DECISIONS

- We define an objective function that uses the predictive models of the two applications.
- Experiment with different objective functions to represent best system performance, and lowest energy.
  - Minimize the sum total of cycles on the machine
  - Minimize the time to completion for the set of benchmarks
  - Minimize energy based on a simple energy model
- We can give weights to the model outputs and other features.

- We use the active-set algorithm for nonlinear constrained optimization (fmincon in Matlab) to solve the objective function.

DECISION-MAKING RESULTS

- We run all possible allocations for the two benchmarks executing together.
- Compare with simple baselines
  - Best Spatial Partition
  - Time-Multiplexing each application on the whole machine
  - Dividing the Machine in Half Spatially

This graph shows how effective our models are a picking an allocation vs. the best and worst alloc. Time-mux'ing or dividing the machine in half.