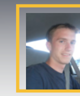
 

Spatial Resource Allocation Using Predictive Models

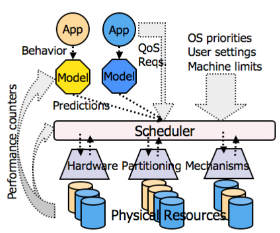
Henry Cook and Sarah Bird

PROBLEM STATEMENT

Chips are becoming increasingly parallel, meaning that more scheduling decisions have to be made. We most 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 in system management. 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 will have on performance, and to use these predictions to make allocation decisions. This plan requires us to create models that capture the relationship between allocations and performance.

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 can 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.

MODEL FORMULATION

We create models from samples of performance data, and use them to predict the performance of allocations not included in the original sample. The inputs to our models are performance and activity metrics. The outputs of our models are predictions of metric values for untested allocations.

Linear additive model:

$$y(x) = a_0 + \sum_{i=0}^{N} a_i x_i$$

Multivariate response surface model:

$$y(x) = a_0 + \sum_{i=0}^{N} a_i x_i + \sum_{i<j}^{N} a_{ij} x_i x_j + \sum_{i=0}^{N} a_{ii} x_i^2 + \ldots$$

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. We can give weights to the model outputs and other features. We use the active-set algorithm for nonlinear constrained optimization (fmincon in Matlab).

METHODOLOGY

We use Virtutech Simics with custom modules supporting hardware partitioning to collect performance data to create the models. We created synthetic benchmarks with varying types of resource requirements to explore the space of possible behaviours.

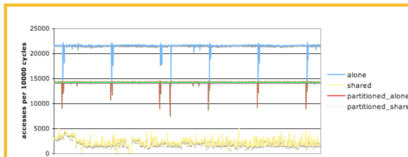
SYNTHETIC BENCHMARKS

Name	Processor	Cache	Offchip BW	Description
pbcbbb	benefits	benefits	benefits	Copies data from many large blocks, with intra-block reuse (multithreaded)
pocbbb	oblivious	benefits	benefits	Copies data from many large blocks, with intra-block reuse (single threaded)
pbcobb	benefits	oblivious	benefits	Streaming copies with no reuse (multithreaded)
pocobb	oblivious	oblivious	benefits	Streaming copy with no reuse (single threaded)
pbcbbo	benefits	benefits	oblivious	Copies data repeatedly from single large blocks (multithreaded)
pocbbo	oblivious	benefits	oblivious	Copies data repeatedly from a single large block (single threaded)
pbcobo	benefits	oblivious	oblivious	Pointer chases through long lists (multithreaded)
pocobo	oblivious	oblivious	oblivious	Pointer chases through a long list (single threaded)

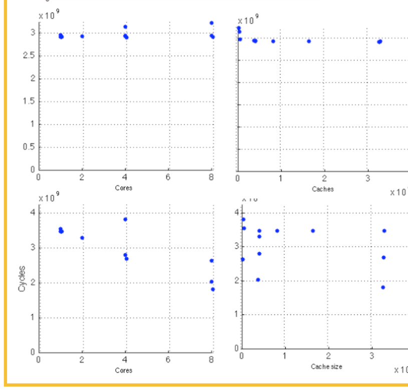
FUTURE WORK

- Retrain models on different sample sizes
- Try making decisions with heuristic search
- Improve realism of energy model
- Make temporal as well as spatial decisions

OBSERVED PERFORMANCE



The top graph shows that our GSF mechanism succeeds in partitioning BW.



These graphs show the effect of varying core count and cache capacity on runtime (in cycles) for two different benchmarks. The upper benchmark's performance is oblivious to the size of its allocation, while the lower one's performance improves with more cores allocated.

MODEL ACCURACY

Name	Cycles	Instructions	Off Chip Accesses	Cache Access
pocobo	3.58 (3.74)	0.22 (0.14)	74.68 (90.86)	12.82 (10.74)
pocobb	21.61 (10.88)	0.85 (0.95)	54.8 (33.01)	91.23 (82.21)
pocbbo	24.98 (21.04)	0.33 (0.35)	54.11 (37.16)	45.74 (40.21)
pocbbb	11.27 (11.08)	0.14 (0.14)	33.58 (31.66)	23.61 (31.17)
pbcobo	46.85 (27.95)	17.53 (10.25)	275.17 (481.34)	48.89 (26.91)
pbcobb	21.51(12.30)	4.21 (11.83)	59.08 (35.30)	105.80 (98)
pbcbbo	20.93 (20.87)	0.32 (0.24)	39.88 (40.73)	34.15 (35.43)
pbcbbb	23.20 (22.05)	0.35 (0.45)	50.53 (37.38)	35.50 (34.18)

Table shows the mean accuracy (standard dev. accuracy) of the response surface model. Some metrics of performance were much easier to predict than others. Outliers severely degrade mean accuracy.

DECISION-MAKING RESULTS

ID	Name A	Name B	Cores A	Cache A	OffchipBW A	Cores B	Cache B	OffchipBW B
1	pocobo	pbcbbb	4	16384	2112	4	16384	3036
2	pbcobo	pocbbb	1	512	300	7	32768	4033
3	pocbbo	pbcobb	1	4096	300	7	24540	4,197
4	pocobb	pbcbbo	1	16384	1000	7	16384	3545
5	pocobo	pocobo	4	16384	2033	4	16384	2034
6	pbcbbb	pbcbbb	4	16384	3079	4	16384	3072
7	pbcbbb	pbcbbo	1	16384	1009	7	16384	4026
8	pbcbbb	pbcobb	1	16384	1000	7	16384	4036
9	pbcbbb	pbcobo	7	24540	4022	1	2048	500
10	pbcbbb	pocbbb	1	512	300	7	32768	4033
11	pbcbbb	pocbbo	7	24540	4022	1	2048	500
12	pbcbbb	pocobb	1	19,386	1,030	7	13,382	4,022

Decisions made when allocating resources between two benchmarks. In red cases, the decisions made based on the models are counter to what we expected. In orange cases it was unclear whether the decisions made were correct or not. Model inaccuracy results in poor decisions