Application Modeling and Hardware Partitioning Mechanisms for Resource Management

Sarah Bird, Henry Cook, Krste Asanovic, John Kubiatowicz, Dave Patterson
ParLab Arch and OS Groups
Resource Allocation Objectives

- Each partition receives a **vector of basic resources** dedicated to it
  - Some number of processing elements (e.g., cores)
  - A portion of physical memory
  - A portion of shared cache memory
  - A fraction of memory bandwidth
- Allocate minimum resources necessary for each applications QoS requirements
- Allocate remaining resources to meet some system-level objective
  - Best performance
  - Lowest Energy
- Doesn’t require application developers to worry about low-level resources
System-wide Adaptation Loop

- **Admission Control**
  - Cell-Creation & Resizing Requests from Users
  - Major Change Request
- **Resource-Allocation and Adaptation Mechanism**
  - Global Policies / User Policies and Preferences
  - Offline Models and Behavioral Parameters
- **Online Performance Monitoring, Model Building, and Prediction**
- **Policy Service**

**Partition Mapping and Multiplexing Layer**
- STRG Validator & Resource Planner
- Partition Implementation

**Partition Mechanism Layer**
- QoS Enforcement
- Channel Authenticator

**Partitionable Hardware Resources**
- Memory Bandwidth
- Cache/Local Store
- Physical Memory
- Cores
- Disks
- NICs
- Performance Counters

**Major Change**
- Request
- Channel ACK / NACK

**Minor Changes**
- Cell-Creation & Resizing Requests from Users
- Partitioning

**Space-Time Resource Graph (STRG)**
- All system resources
- Cell group with fraction of resources
- Cell

**Offline Models and Behavioral Parameters**
- Resource-Allocation and Adaptation Mechanism

**Online Performance Monitoring, Model Building, and Prediction**
- Performance Counters
Techniques and Tradeoffs

APPLICATION MODELING
Motivation

• Programmers are unlikely to know exactly how low-level resources effect performance
  – Developers are concerned application-level metrics
    • e.g., frames/sec, requests/sec
  – Operating system has to make decisions about resource qualities
    • e.g., number of cores, cache slices, memory bandwidth

• Automatically constructing performance models is a good way to bridge the gap between application-level metrics and hardware resources
Model Building

• Collect data points of performance for specific resource allocation vectors
  \[ L_i = PM_i(r_{(0,i)}, r_{(1,i)}, \ldots, r_{(n-1,i)}) \]
• Use multivariate regression techniques to fit a model to the data points

<table>
<thead>
<tr>
<th>Linear and Quadratic</th>
<th>GPRS</th>
<th>KCCA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td><strong>Advantages</strong></td>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>• Simple to build</td>
<td>• Very Accurate</td>
<td>• Can represent all output metrics in one model</td>
</tr>
<tr>
<td>• Work well with simple optimizers</td>
<td></td>
<td>• Successful in the past</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td><strong>Disadvantages</strong></td>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>• Potentially inaccurate</td>
<td>• Can overfit the data</td>
<td>• Doesn’t work with simple optimizers</td>
</tr>
<tr>
<td>• Can’t represent variable interaction</td>
<td>• Computationally expensive to build</td>
<td></td>
</tr>
</tbody>
</table>
Model Accuracy
Model Creation
Online vs. Offline Training

• Offline profiling options
  – Profile applications in advance
    • Distribute with application
    • iTunes App Store or Android Market
  – Create application profiles in the Cloud
    • Record performance and resource statistics from users
    • MSR is currently doing this to make perf. models for app developers

• Online profiling options
  – Install time profiling
    • Operating system tests out a variety of configurations
  – Online refinement of models
    • Operating system starts with a generic model
    • Retrains the model with new information as the application runs
Performance Isolation

• Without performance isolation,
  – An applications performance could vary widely as a result of concurrently running applications
  – Inaccurate models
  – Requires different models of the application based on the system load

• Performance predictability is an important component for application modeling
  – Other advantages
    • Better tuned, more efficient applications
    • Easier to make QoS guarantees
RESOURCE ALLOCATION USING CONVEX OPTIMIZATION

An Example
Minimizing the Urgency of The System

[Burton Smith (MSR), Operating System Resource Management (Keynote), IPDPS 2010]

Minimizing the urgency of the system can be formulated as a continuous optimization problem subject to restrictions on the total amount of resources. The urgency function, \( U(L) \), is related to the performance model, \( PM(r_i) \), which describes how the performance of the system changes with respect to resource allocation.

\[
L_a = PM_a(r_{0,a}, r_{1,a}, \ldots, r_{n-1,a})
\]

\[
L_b = PM_b(r_{0,b}, r_{1,b}, \ldots, r_{n-1,b})
\]

\[
L_i = PM_i(r_{0,i}, r_{1,i}, \ldots, r_{n-1,i})
\]

The objective is to continuously minimize the urgency function, \( U_a(L_a) \), \( U_b(L_b) \), and \( U_i(L_i) \), subject to constraints on the total amount of resources.
Urgency Function

[Burton Smith (MSR), Operating System Resource Management (Keynote), IPDPS 2010]

• Reflects the importance of cell $C_i$ to the user

$$U_i(L_i) = \text{MAX}(s_i \cdot (L_i - d_i), 0)$$

\[L_i = \text{PM}(r_{(0, i)}, r_{(1, i)}, \ldots, r_{(n-1, i)})\]

$r_{(1,i)}$: Allocation of resource of type 1 to Cell $C_i$
Performance-Aware Convex Optimization for Resource Allocation

[Burton Smith (MSR), Operating System Resource Management (Keynote), IPDPS 2010]

- Advantages
  - Convex optimization is relatively inexpensive optimization problem with a single extreme point
  - Urgency Function Slopes allow the system to express relative priorities of application
  - Priorities change as a function of performance
  - Urgency Function Intercept encapsulates QoS requirements
  - And additional process can be used to represent system energy

Very sensitive to the performance models of the applications
Approaches + Mechanisms

HARDWARE PARTITIONING
GSFm: Globally Synchronized Frames
Bandwidth Partitioning for the memory hierarchy

- Frame-Based QoS System
  - Transactions are labeled with a frame number
  - Head frame moves through the network with a top priority

with Jae Lee, MIT
GSFm: Globally Synchronized Frames
Bandwidth Partitioning for the memory hierarchy

- Uses source-side suppression
- Applications are given a bandwidth allocation per frame
  - Credits per resource
    - Memory channels and memory bank, network link, etc
  - Memory transactions are charged for all possible resources
    - Delayed into a future frame if the app doesn’t have enough credits

<table>
<thead>
<tr>
<th>Networks</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2hREQ</td>
<td>h2cRESP</td>
</tr>
<tr>
<td>h2cREQ</td>
<td>c2cRESP</td>
</tr>
<tr>
<td>channel</td>
<td>bank</td>
</tr>
<tr>
<td>1H</td>
<td>1P</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1T</td>
<td>1T</td>
</tr>
</tbody>
</table>

H: header-only message
P: header+payload message
T: memory transaction
Advantages of GSFm

• Minimum Bandwidth Guarantees
  – Flexible
  – Differentiated
  – Weighted sharing of the excess bandwidth

• Good Utilization
  – Early frame reclamation
  – Excess bandwidth

• Minimal Hardware Requirements
  – Reasonable area
  – Distributed
Cache Partitioning

• Way-Based
  – Simple Indexing
  – Changes the replacement policy
  – Reduced associativity
  – Limited by number of ways
  – No locality for NUCA systems

• Bank-Based
  – Locality in NUCA
  – More complex indexing
  – Requires flush on reconfiguration
  – Limited by number of banks
RAMP Gold Experiments

A SIMPLE EVALUATION
Experimental Platform

• RAMP Gold: FPGA-Based Simulator Target Machine
  – 64 single-issue in-order cores @ 1GHz
    • Partitionable into sets of 8
  – Private L1 Instruction and Data Caches each 32KB
  – Shared L2 Cache 8MB inclusive 10 ns latency
    • Partitionable into 8 slices using page coloring
  – Memory bandwidth with magic interconnect
    • Partitionable into 3.4 GB/s units assigned to a set of cores

• ROS Kernel Code
  – Microbenchmarks & PARSEC Benchmarks
Application Modeling

• Use 10 sample points
  – 18.5% of the 54 Possible Allocations
  – Selected using Audze-Eglais Design of Experiments

• Create a model of the application
  – Input: Resources
  – Output: Performance

• Explore different types of models
  – Linear
  – Quadratic
  – KCCA (Machine Learning)
  – Genetically Programmed Response Surfaces (GPRS)

• Run all 54 Allocations to test model accuracy
Scheduling Experiment

• Evaluate Objective Function for a Pair of Benchmarks using the models
  – Race to Halt
    • Min Max(Cycle1, Cycle2)
  – Least Cycles
    • Min (Cycle1*Cores1 + Cycle2*Cores2)
  – Lowest Energy
    • Min $\Sigma_i$ (resource utilization $i$ * energy parameter $i$)
  – Using MATLAB’s fmincon

• Run all 54 possible resource allocations for each pair of benchmarks
  – Assumes that all resources must be allocated
• Time-Mux’ing is on average of 2x worse than the best spatial partition

• However the worst spatial partition is quite bad.

• Naively dividing the machine in half is 1.75x worse than the best spatial partition

• Linear model is within 8% of optimal every time

• Quadratic Model is within 3% of optimal every time

• KCCA does well in some cases but very poorly in others
Conclusions

• Simple application models show promise
  – Still lots of challenges
    • When to build?
    • How to store?
    • Variability

• Resource allocation using convex optimization potentially very interesting
  – Lots of parameters to tune
    • How do we set the urgency functions?
  – How does it compare with other options?
QUESTIONS?