Inadequacy of Static Mapping

Collision of two trends

- Increasing diversity in execution platforms
  - uniprocessor
  - parallel/distributed
    - CPU capabilities, memory capacities, network characteristics

- Applications span multiple execution platforms
  - a component needs to interact with other components that run on diverse platforms
  - a component must itself run on multiple platforms

Consequence

- Larger penalties for bad mapping decisions

Diversity: Uniprocessors

Trend 1:
a widening memory gap

<table>
<thead>
<tr>
<th></th>
<th>10 GB/s</th>
<th>500-1000 cycles</th>
<th>~1 GB/s</th>
<th>100-1000 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 GB/s</td>
<td>1-10 cycles</td>
<td></td>
<td>~70 MB/s 100-1000 cycles</td>
</tr>
</tbody>
</table>

**Computer System Layers**

- Mapping of program components to physical resources is typically controlled by static OS policies
  - (sometimes), these policies can dynamically change the mapping
    - e.g., virtual-memory page replacement policies
  - (rarely), these policies can adjust themselves dynamically
    - e.g., switch from one page replacement policy to another
  - (very rarely), the application can control some policy parameters
Diversity: Parallel/Distributed Systems

- WAN, Wireless, LAN, SAN...
- differ in latency
- BW error properties
- QoS attributes (e.g., jitter, loss)

PDAs, Mobile PCs, Servers, SMPs...
- differ in CPU performance/capabilities
- memory capacity
- internal bus BW
- display attributes

Application Requirements

- **Uniprocessors:**
  - same application must execute in diverse environments
  - binary must
    - take advantage of available hardware
    - configure hardware as appropriate for the application
    - tradeoff different resources (e.g., memory access vs. computation)

- **Parallel/Distributed platforms:**
  - application components must interoperate across an even larger range of computing and networking capabilities
    - e.g., different clients talking to a server component
  - in emerging applications, the same component may need to migrate across multiple such platforms
    - e.g., mobile code/applets
  - a harder problem because environment can change at run time

Computer System Layers: Modern View

- **Controlled departure from layered model**
- **Advantages**
  - application/OS is better able to utilize hardware based on application requirements
  - application/OS can adapt to changes in resource characteristics (if any) in an application-specific fashion
  - improved performance, reliability, etc.

Course Objectives

- Develop an understanding of the state-of-the-art research on
  - architectural and OS adaptation mechanisms
    - what “hooks” are available to control the mapping?
  - application-level adaptation frameworks
    - how should applications be structured to permit adaptation?
  - (not the main focus) adaptation policies
    - how to adapt?

- Use this understanding in the course project to develop an adaptation strategy for a specific distributed application
A Restricted View of Adaptation

- Emphasis on application-driven adaptation
  - adaptation "logic" is influenced by application semantics

- We shall not discuss
  - adaptation behaviors hard-coded into the architecture/OS
    - architecture adaptation
    - branch prediction
    - cache management
      - e.g., some cache-coherence schemes in SMPs that dynamically change from update-based to invalidation-based
    - OS adaptation
      - ack window adjustment for TCP congestion-control
      - page replacement policies that change their behavior
  - these are examples of application-oblivious adaptation
    - very useful, but not the focus of this course

Course Schedule

Lectures 7-9: Adaptation Frameworks
Lectures 5-6: Monitoring, Extensible OSes
Lectures 2-4: CPU, Memory System, Secondary Storage, Network Interfaces, Network

Course Structure

- Different courses require different approaches
  - undergraduate: Instructor lectures, students listen
  - beginning graduate: Instructor lectures, students listen + investigate on their own
  - advanced graduate: Instructor is just another student
    - discussion format is most suitable
    - we are all going to learn something from the course

- Three components
  - Paper presentations
  - Class discussion and Summary writeups
  - Project

Grading Criteria

- Presentations - 10%
  - what are the mechanisms? how much do they cost? what are the potential benefits?
  - what are the elements of the framework? is the burden on the programmer reasonable? what are the run-time costs of the framework?

- Discussion - 10%

- Three writeups (less than 10 pages) - 30%
  - architectural adaptation mechanisms
  - OS adaptation mechanisms
  - application adaptation frameworks

- Project - 50%
**Project**

◆ **Context**
  - a distributed visualization application
    - clients download images from a server
    - server loads images from disk, processes them, and sends them to clients
    - clients receive data from network, process the image, and display it
  - an execution platform capable of supporting programmable adaptation mechanisms
    - local node interactions via a virtual node environment
    - network interactions via the Myrinet programmable NIC

◆ **Challenge**
  - design an adaptation strategy (mechanisms, policies, framework) that will deliver robust application performance over a range of system load scenarios

◆ **Start as soon as possible!**

**Next Lecture**

◆ **Architectural Mechanisms for Adaptation**
  - Architectural Adaptation for Application-Specific Locality Optimizations, Zhang et al.
  - PipeRench: A Coprocessor for Streaming Multimedia Acceleration, Goldstein et al.

◆ **Volunteers?**