Tracing in Parallel Systems

Why Tracing?
- Parallel programs are complex
  - many processing tasks running concurrently
  - complex synchronization patterns
  - computation vs. waiting for synchronization event
- Tracing
  - record events and timing during execution
  - provide log data or even visualization for later analysis
- Where is the tracing done?
  - application software
  - system software

MPICH Tracing
- Compile mpicc with -mpilog flag
- Execute to produce log file (clog or slog)
- jumpshot to interactively visualize the traces

Tracing for Shared-Memory Parallel Programming
- What kinds of events you want to trace?
- Where is the tracing done?
### Tracing in Concurrent Server

**Goals:**
- Debug performance problems
- Measure performance and resource accounting at each request
- Request classification and anomaly detection

**Challenges**
- Highly multiprogrammed, many context switches
- Request processing is multi-stage - web server, J2EE components, database

**Where is the tracing done?**
- Applications
- Programming systems
- Operating systems
- ... ...

### Requests in Server

**What is a “request”?**
- From application point of view, the task the server fulfills on behalf of a user demand:
- From system point of view, the set of all activities (in the server system) in fulfilling a single application request

**Tracking individual request**
- Microscopic view of the system
- Allows performance management and resource accounting at request granularity

### An Example of Request Energy Usage Distribution

**Track Request CPU Usage**

- Oftentimes a request maps to a process/thread
- How to track a process CPU usage in OS?
  - Sampling vs. tracking context switches
  - Accounting for interrupt handlers
- Exceptions that a request does not exactly map to a process
  - Process pooling
  - Multi-stage server
  - Background tasks
Resource Containers
[Banga et al. OSDI1999]
- Tracking requests
  - Applications pass request handles when requests go from one stage to the next
- Request-granularity resource management
  - What is the default scheduling unit in operating systems?
  - Decouple scheduling context, protection domain, and resource allocation context
  - Flexible/better resource control - e.g., give certain users higher priority or guaranteed CPU allocation

Magpie [Barham et al. OSDI2004]
- Tracking requests transparently at the OS
  - Dump bunch of system events
  - Analyze the events afterward to find out request context propagations
    - process A sends something into a socket and process B reads out of the same socket a bit later
    - an interrupt handler is linked to a page cache entry which is later read by process C
- Flexibility
  - A framework with user-defined rules
  - Extending this into application level
- Not in real time

Utilization of Magpie Request Tracking
- Request classification
  - Each request is identified by its events
    - e.g., a chain of system call IDs
  - Put similar ones into groups/clusters
    - what is the good metric to quantify the difference between requests?
- Request modeling
  - Given an online workload, you can say
    - 80% are like this ...
    - 19% are like this ...
    - a couple of really weird ones

Real-Time Request Tracking
[Our work, ASPLOS2008]
- Real-time tracking allows immediate system reaction
  - Adaptive OS management policy
  - Anomaly detection and quarantine
- Most request propagations are through inter-processing communications
  - Tag request contexts to messages
- User-level request context propagation is hard to detect
An Example of Request Context Propagation

---

Diagram showing the flow of request context through various components such as Tomcat, JBoss, RUBiS, MySQL Database, and others.