

Slice, Proportional Share, and Gang Scheduling

- Last Time: Batch Scheduling Resource Model
 - » High Resource Efficiency & Indeterminate Wait
 - » Complications: Resource Requirements; Parallel Jobs, Advance Reservations
 - » Performance: pretty high resource utilization
 - » Prediction
- Today
 - » Slice and Proportional Share Scheduling
 - » Gang Scheduling and Coscheduling
 - » Best Effort Slice in Planetlab

CSE225 – Lecture #11

Today's Readings

- Waldspurger and Wehl, Lottery Scheduling: Flexible Proportional-Share Resource Management
- Choi, Kim, Ersoz, Yoo, and Das, Coscheduling in Clusters: Is it a Viable Alternative?, Proceedings of Scalable Computing and Networking (SC2004), 2004.
- Bavier, et. al. Operating System Support for Planetary-Scale Network Services
 - » Qie, et. al. Scheduling Computations on a Software-Based Router

CSE225 – Lecture #11

Resource Management Problem

- Application Perspective:
 - » Given my application, find and bind an appropriate (“best”, “acceptable”, “best below acceptable”) set of resources
 - » => Optimize for application quality or performance
- System Perspective:
 - » For a set of resources, identify a set of applications which make good use of the resources (“best”, “acceptable”, “high utilization”, etc.)
 - » => Optimize for system utilization or “total value”

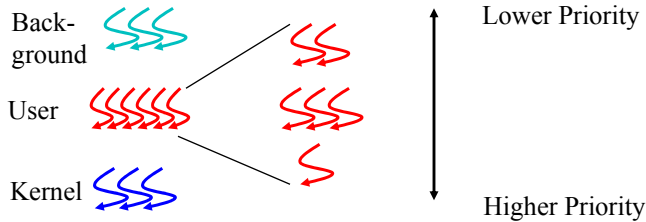
CSE225 – Lecture #11

Resource Management Models

- “Cycle Stealing”
 - » Volatile, Asynchronous Preemption
- Dedicated Resource Scheduling
 - » Batch schedulers, dedicated resources, advanced reservation
- Time-sharing **TODAY**
 - » Slice schedulers, proportional share, guaranteed/best effort
- Objective: Understand implications for distributed application performance and motivations/advantages of various models (reach, local)

CSE225 – Lecture #11

Timesharing Schedulers



- Classical Unix Priority-decay schedulers
 - » Multiple priority levels; Round Robin within a priority
 - » Priority = (recent CPU usage / 2) + base user priority
- Periodic Priority Adjustment: Design for Responsiveness
 - » Increase the priority of I/O intensive jobs
 - » Degrade the priority of compute intensive jobs
- Roughly fair, but significant inequities can occur

CSE225 – Lecture #11

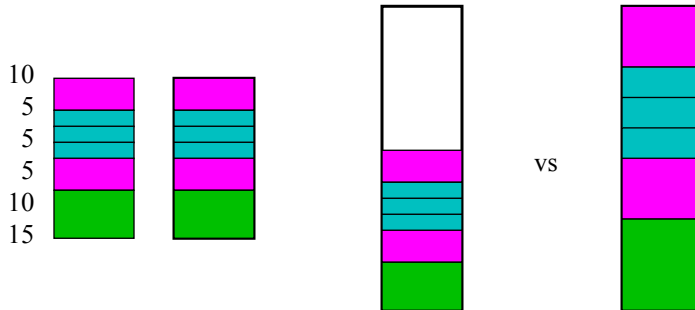
Proportional Share Scheduling



- Based on Weighted Fair Queueing (WFQ): Long-term stronger fairness guarantees
- Elements
 - » Each process reserves a capacity (compute rate – Mcps, not an application metric)
 - » Unused and Unallocated capacity is distributed across users in proportion to their allocation
 - Idle processes do not accumulate their credits – they are lost
 - » Guarantees *isolate* the performance of the processes from each other

CSE225 – Lecture #11

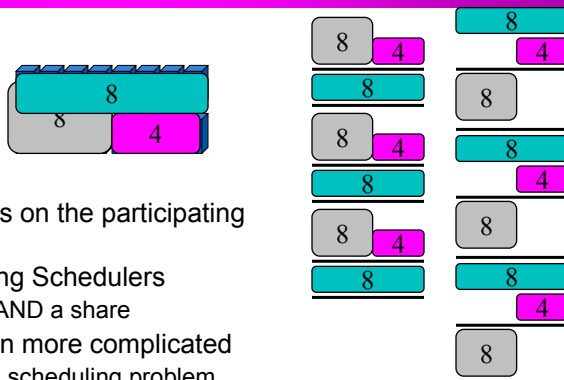
Proportional Share Scheduling, Excess and Shortage



- Allocated Capacity => Delivered Capacity
- Excess Capacity? Or not used?
- Overallocation: not allowed, fail to meet shares
- Admission control and Queuing; Good load range

CSE225 – Lecture #11

Coordinating Schedulers: Gang Scheduling



- Coordinate the schedulers on the participating nodes
- Mix Batch and Timesharing Schedulers
 - » Allocate a set of nodes AND a share
- Scheduling becomes even more complicated
 - » Adds a dimension to the scheduling problem (which fit together in a partition size)
- Works well for development; at what granularity
 - » Reduces queueing delay
 - » Doesn't work for memory-intensive jobs

CSE225 – Lecture #11

Evaluation of Co-scheduling

Coscheduling Scheme	Receiver	Sender	Collective Communication	
			Beginning	End
DCS	Spin-only, Boost a priority	Spin-only	Nothing	Nothing
SB	Spin-Block	Spin-only	Nothing	Nothing
PB	Spin-only, Boosting	Spin-only	Nothing	Nothing
CC	Spin-Block, Boost a Priority	Spin-Block, Boost a Priority	Nothing	Nothing
HYBRID	Immediate Block	Immediate Block	Boost a priority	Restore

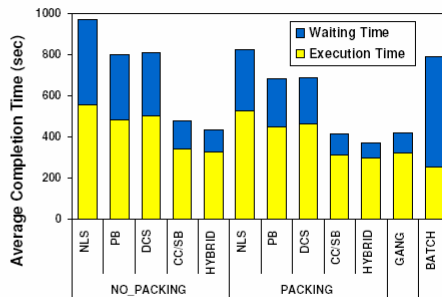
- Gang-scheduling is too Rigid, explore overlap, unstructured techniques
- Communication Driven Techniques
 - » Dynamic Co-scheduling (CS)
 - » Hybrid (improve CS, smart on block, priority)
- Gang (master controlled co-scheduling)
- Batch (dedicated machine baseline)
- Packing: Contiguous Allocation
- No Packing: Random Allocation of # nodes

CSE225 – Lecture #11

Evaluation of Co-scheduling



(a) High Load

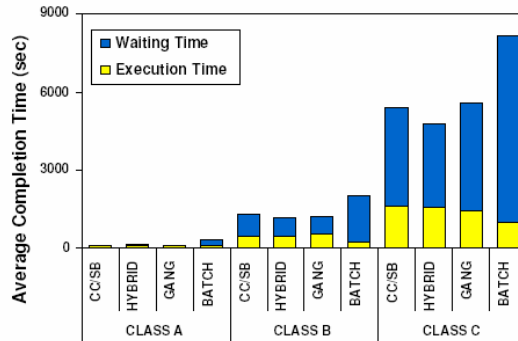


(b) Light Load

- 16-node clusters, NAS Parallel Benchmarks Workload, 100 jobs
- Waiting time vs. execution time (on the machine)

CSE225 – Lecture #11

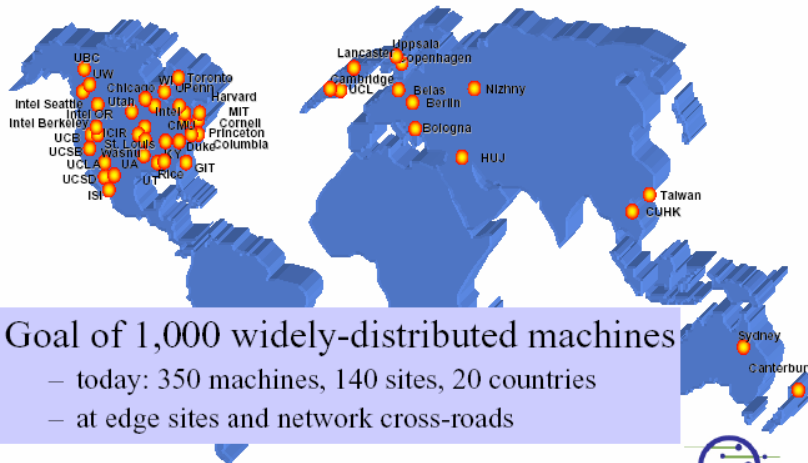
Evaluation of Co-scheduling



- Vary Problem Sizes

CSE225 – Lecture #11

What is Planetlab?



1/29/2004

10

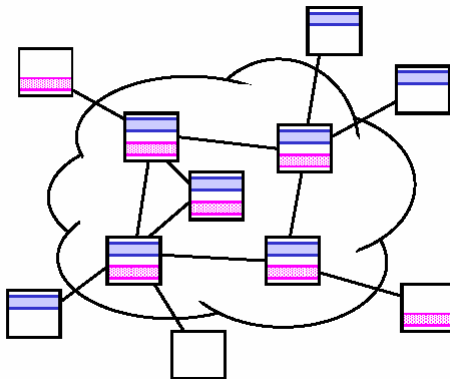


PlanetLab is...

- A form of a Grid (shared open infrastructure)...
- Designed to enable innovation in planetary-scale services...
 - » Multiple vantage points: anomaly detection, robust routing
 - » Proximity to source/sinks: content distribution, data fusion
 - » Multiple independent domains: survivable storage
- More focused on next generation internet structures, services, etc.
- Less focused on computing and storage
- =>But really hard to separate the two

CSE225 – Lecture #11

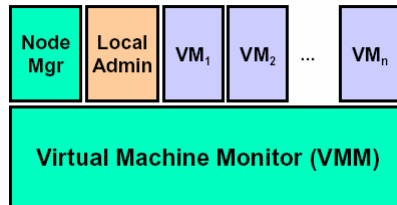
Slices – Distributed Virtual Machines



- Planetlab -- an OS image on each node (private filesystem and network namespace)
- Slice Extends the notion of a “virtual machine monitor” to distributed resources, connected by a network

CSE225 – Lecture #11

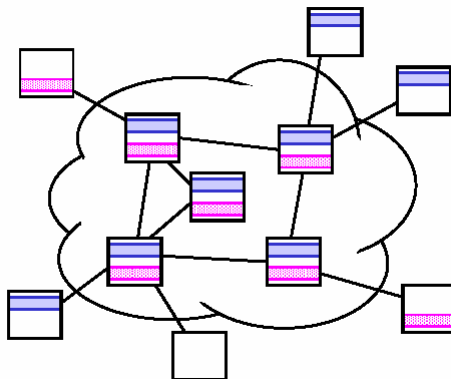
Per Node View



- Local Scheduling based on Proportional Share
- Slices are “pseudo-static” so, don’t overallocate or queue

CSE225 – Lecture #11

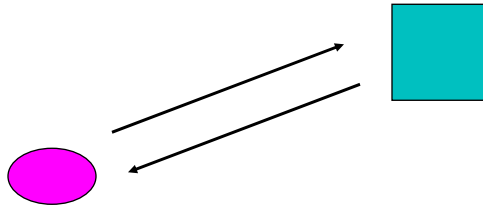
Distributed Scheduling



- Coordinated (Gang) and Uncoordinated Scheduling (Independent)

CSE225 – Lecture #11

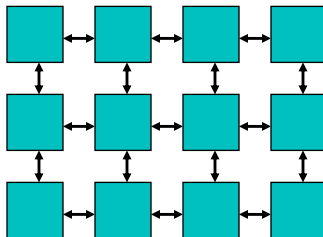
Performance Effects: Client-Server



- Network latency (variable) combines with scheduling latency (variable)
- Fairness ensure eventual response
- Load: Better at low, degrades at high
- Distance: Okay at distance, degrades close

CSE225 – Lecture #11

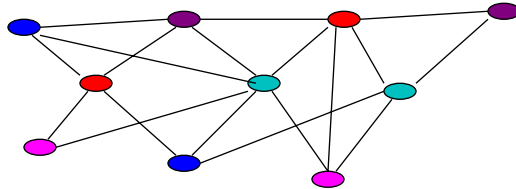
Performance Effects: Parallel Jobs



- Variable latencies can slow the entire computation, due to tight coupling (compound and propagate?)
 - » Load: Better at low, degrades at high
 - » Distance: Okay at distance, degrades close
- Asynchronous or loosely-coupled applications will behave like client-server

CSE225 – Lecture #11

Performance Effects: Peer to Peer / Proxy Networks and Applications



- Multiple hops in CDN (find), multistage lookup or fanout
 - » Small additive latency, may not be significant
- Data Transfer direct access, pipelined forwarding, multi-path
 - » Memory intensity?
- Highly adaptive to network or resource responses (load migrates)
- Others?

CSE225 – Lecture #11

Discussion: Open Questions and Ideas

- How could you include slice scheduled resources into a distributed application?
- How does asynchronous scheduling affect what types of distributed coordination are possible? latencies?
- Interaction of high performance comm and asynch scheduling?
- What are the implications of slices? What type of information would you collect to do a resource utility study?
- How would you compare batch scheduling and proportional share scheduling as a basis for distributed or grid applications?
- How are the applications we discussed (way back) suited to aggregation of these types of resources?

CSE225 – Lecture #11

Comparing Resource Management Models for Distributed Applications

- Cycle Stealing: Asynchronous Departure
- Dedicated Resource Scheduling
- Time-sharing, Proportional schedulers

CSE225 – Lecture #11

Discussion

- How do these compare as a basis for distributed/grid applications?
- Which are promising and why? Under what conditions?
- Are new models possible? Desirable?
- What are the fundamental limits and challenges?

CSE225 – Lecture #11

Summary

- **Local Scheduling: Timesharing to Proportional Share**
 - » Long-term Fairness, Limited Capacity and Admission Control, Allocation of Excess and unused
- **Gang Scheduling: Combining Batch and Timesharing**
 - » Coscheduling: generalized view and benefits
- **Slices: Distributed Virtual Machine Monitors and Networks**
 - » Proportional Share scheduling, Uncoordinated and Coordinated
 - » Impact on capabilities and performance
- **Model Implications on Grid Applications**