Job Scheduling in Multiprogrammed Parallel Systems

by Dror G. Feitelson

- Parallel systems/many users/competing programs
- Processor scheduling goals
  - Different, for different people
  - Interactive and batch environments
  - Systems people: maximize utilization/throughput of their system
  - End-users: minimize turn-around time of their job
- Hard problem, with many conflicting goals!
- Wide variety of proposed solutions
- No ideal scheduling policy exists for all user requirements

- Resource allocation – NOT program development
- FOCUS
  - single level
  - two level
  - time sharing
  - space sharing
- Other types of scheduling
  - mapping problem
  - real time constraints
  - metacomputing
  - NOWs and interaction of parallel and sequential workloads

- Classic distributed or shared memory parallel machines
- Terminology
  - “thread” as process or lightweight process
  - job, application, program
  - workload
  - multiprogramming *vs* multitasking
  - scheduling, allocation, partitioning ...
- **Classification**
  - Single CPU systems
  - Complexity increases in parallel systems
    - ◇ single level (problems...)
    - ◇ two level (decoupling of problems) space-sharing and time-sharing

Classification of Scheduling Policies

- Two-level scheduling appropriate for shared memory machines but not for distributed memory systems
- Space Sharing: one level
  - programs in SPMD style, a single thread to each PE
  - easy, widespread
  - could be seen as space sharing
- Time Sharing: second level of a two-level policy
  - extension of uniprocessor systems
  - could be seen as pure time sharing
- Gang-scheduling
  - hybrid scheme
  - combines the two other schemes but harder to implement

Overall: Many possibilities

What does work best?

Depends on assumptions...

- Type of system, its use...
  - theoretical studies (how much do we know?)
  - practical ones (usually, we know less...)
- System architecture
  - shared memory
  - distributed memory
- Parallel jobs
  - rigid
  - moldable
  - evolving
  - malleable *most desirable*

What does work best? (cont.)

Depends on performance metrics...

- Application level scheduling *vs* “whole” system performance
- Popular system metrics:
  - System throughput (maximize system throughput before saturation)
  - Average job response time (minimize the average job turn-around time)
  - System power (maximize $\frac{Thoughput}{ResponseTime}$)
- Fairness (long versus short jobs)
- User satisfaction
- Quality of service (lack of surprise)

Parallel Workload Abstraction

- Characterization studies
- Single application characteristics (e.g., application scalability)
- Composite characteristics of *all* applications in the system
  (e.g., inter arrival rates, multi class)
- Stochastic distributions

Single Application Characteristics

- Models of parallel application scalability
- Speedup $Sp(n) = \frac{T(1)}{T(n)}$ and its various functional forms
- Efficiency $fracSp(n)n$
- Average Parallelism
- Maximum Parallelism
- Parallelism Variance
- Processor Working Set
- Parallelism profiles and parallelism shapes

... such abstractions have guided early policy developments ...

Composite Parallel Workload Characterization

Recent studies (since 1995...)

- Production parallel workload characteristics at the NASA AMES iPSC/860 [Feitelson Nitzberg '95]
- Comparative study of the iPSC/860 requirements with the Paragon at SDSC [Windisch *et al* '96]
  - similarities: job size distribution, inter arrival distribution
  - differences: diversity in the mix of jobs, system utilization
- Workload evolution at CTC SP-2 [Hotovy '96]
- Modeling of parallel job lifetime distribution [Downey '97]

Composite Parallel Workload Characterization

- Similarities: the coefficient of variation of parallel job inter arrival distribution and parallel job execution time is > 1
- Popular exponential assumptions do not hold

But let's talk about specific policies: Space Sharing

Definition: One job per partition

- Why space share?
  - reduces O.S. overhead
  - avoids (as much as possible) virtual memory paging
  - cheap, easy to implement
- Desirable features
  - Independence
  - Flexibility
  - Adaptiveness
  - Cost and complexity
  - Modularity and scalability

- Limitations
  - hardware (e.g., interconnection topologies)
  - software (application size)
- How and where to place partition boundaries?
- Classification of space-sharing
  - fixed partitioning (static)
  - variable partitioning (static)
  - adaptive
  - dynamic

Advantages/Disadvantages

Fixed Partitioning

- Equal size partitions
- Variable size partitions
- Easy, easy
- Not efficient
- Adaptivity issues
  - system load
  - workload scalability
- Easy to model (think of a single class case)
- Can be seen as an $M/M/c$ queue
- Can you predict how performance measures will look like?
- Fragmentation

Variable Partitioning

- According to requests
- Restrictions imposed by architecture
- No internal fragmentation, but certainly external
  - hypercubes: partitions based on powers of 2
  - multistage networks
  - mesh-based machines
    - 2-d version of the buddy system (but limited to square systems where the side is a power of 2).
    - not necessarily rectangle allocation
    - fragmentation
- How many processors to ask? (think of queuing, efficiency, parallelism...)

Old Solutions to a New Problem

- Motivation: bin-packing
- Shortest Job First
  - drawbacks...
- Smallest Job First
  - drawbacks...
- Smallest Cumulative Demand First
  - not great improvements from smallest job first...
- Overall
  - tend to saturate faster
  - FCFS is superior

Adaptive Policies

- More flexible than static policies
- Partition size is computed before the job starts execution
- Partition size is a function of the load in the system (as reflected by the waiting queue length)
- Partition size is a function of the workload parallel characteristics (e.g., PWS, average parallelism, minimum and maximum parallelism, parallelism variability)

Adaptive Policies (cont.)

- Processor Working Set policy (PWS) [Ghosal *et al* '91]
  - each job is assigned at most PWS processors
  - many versions (e.g., FIFO, non-FIFO)
  - decrease ("adapt") the partition size as load increases
- Run-to-Completion (RTC) policy [Zahorjan McCann '90]
  - each job is assigned at most its maximum parallelism
  - free processors are equally distributed among jobs

Adaptive Policies (cont.)

- *A+* policies [Sevcik '89]
  - uses detailed information on the workload characteristics
  - partition size is a function of: system load, minimum and maximum parallelism, variance in parallelism
- AVG policies [Chiang *et al* '93]
  - each job is assigned at most its average parallelism
  - SAVG (variation): smallest average parallelism first

Adaptive Policies (cont.)

- Robust Adaptive (RA) policies [Rosti *et al* '93]
  - do not require any knowledge of the workload characteristics
  - "target" partition
  - "split" and "merge" strategies
- Non-Work-Conserving (NWC) policies [Smirni *et al* '95]
  - deliberately leave processors unassigned
  - adapt easily to irregular conditions

Adaptive Policies (cont.)

But...

- Performance benefits of adaptive policies are still limited
- Unable to "quickly" respond to workload changes
- Major oversight: memory considerations...

Dynamic Policies

- Job partition size can be modified *during* the job execution
- Various policies targeting shared memory systems have been proposed
  - Process Control Policy [Tucker Gupta '89]
    - equal partitions
    - application dynamically controls and adjusts the number of its processes
  - Dynamic Partitioning [McCann *et al* '91]
    - processor assignments change depending on the job's instantaneous parallelism

Dynamic Policies (cont.)

- More expensive because of repartitioning costs
  - data and job migration
  - processor preemption/coordination
  - application reconfiguration
- In distributed memory environments, these costs may outweigh benefits [Squillante '95]
- Many "limited" preemption policies have been developed

Dynamic Policies (cont.)

- Limited premption policies [Chiang *et al* '94]
  - two-level policies
  - one preemption is allowed
- Dynamic with a lower bound [Naik *et al* '93]
  - lower bounds on the time between preemptions
- Folding policies [McCann Zahorjan '95]
  - thread placement and load balancing

Time Sharing Policies

Derived from classic uniprocessor scheduling techniques...

- Round Robin Process [Majumdar *et al* '88]
  - single queue for all processes
  - processes are scheduled independent of their job membership for a time quantum
- Round Robin Job [Leutenegger Vernon '90]
  - quantum size of a job is a function of the number of its component processes

Time Sharing Policies (cont.)

- Round Robin A [Zahorjan McCann '90]
  - each process is allocated at most as many processors as its average parallelism A
  - "creating" and "merging" time slices

But there are problems: "Process Thrashing" [Ousterhout '82]

Gang Scheduling

Solution: All runnable processes of a job should execute simultaneously

- Coscheduling algorithms [Ousterhout '82]
  Tasks of all jobs are put in a common linked list, a window slides through the linked list, and tasks that fit in a window execute simultaneously
  - Matrix Method
  - Continuous Algorithm
  - Undivided Algorithm

Gang Scheduling (cont.)

Many variations of coscheduling that effectively *integrate* time sharing and space sharing

- Distributed Hierarchical Control [Feitelson Rudolph '90, '92]
  - flexible partition sizes, flexible quantum length
  - analytical modeling [Squillante *et al* '96]
  - implementation on the SP-2 [Franke *et al* '96]
- Dynamic and Prediction Coscheduling [Sobalvarro Weihl '95]
  - address fragmentation that may occur with traditional coscheduling

Processor-Memory Scheduling

- In distributed memory systems, there is limited memory capacity in each processor
- Most policies decrease the job partition as load increases
- Important trade-off between processor and memory allocation
- Paging can be very expensive
- Memory considerations must be incorporated into existing policies

Processor-Memory Scheduling (cont.)

- Analytic modeling study [Peris *et al* '94]
  - effects of contention caused by paging
  - performance is sensitive to memory considerations
- Simulation study [Setia '95]
  - adaptive partitioning with memory constraints
  - relaxed memory constraints
  - sensitivity analysis

Processor-Memory Scheduling (cont.)

- Theoretical bounds [Parsons Sevcik '96]
  - equi-allocation
  - throughput is bounded by processor and memory availability
  - memory-processor allocation (first fit in memory)
- Memory considerations coupled with dynamic disciplines
  [McCann Zahorjan '95]
  - jobs with minimum resource requirements
  - equity of service
  - optimal policies (under restricted conditions)
  - heuristic-based policies

Processor-Input/Output Scheduling

- Scheduler and file system
- Must operate in concert
- Examples...