Whole Node Scheduling

Eric Vaandering

OSG Summer Workshop

9-11 August 2011

Slides from

José Hernández

Dave Evans

Chris Jones

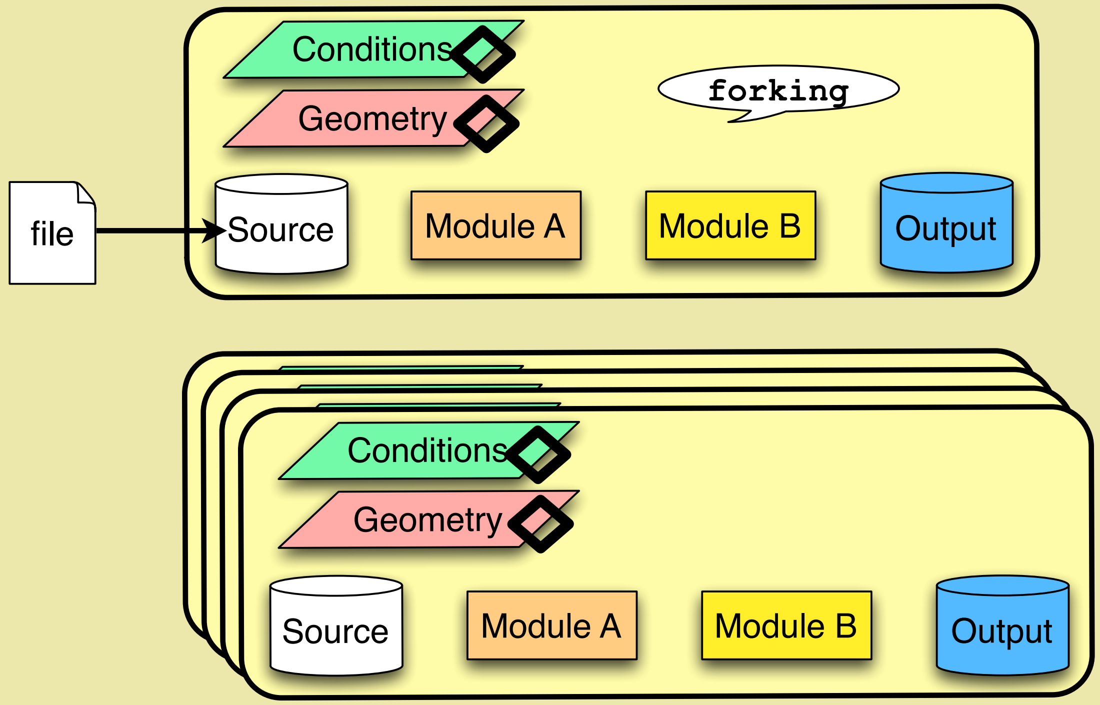
Motivation for multicore processing

- RAM available in WNs is a limitation for production
- Multi-core aware applications can improve memory sharing
- I don’t think this applies to Tier3’s (explain at end)
- CMSSW forking
  - Parent process loads calibrations, conditions, geometry
  - Parent forks children
  - Children share parent (read-only) memory and process a fraction of the input file
  - Execution script merges results
  - CMSSW testing: 13 GB used by 32 children, 34 GB used by 32 separate jobs

Forking in CMS

Parent

- Reads configuration and loads modules
  - Configuration says how many children and # events/child
- Opens input file and reads first run
  - modules are not called
- Pre-fetches conditions, calibrations and geometry
- Sends message to all modules that forking is going to happen
  - source closes file
- Forks



Forking in CMS (cont)

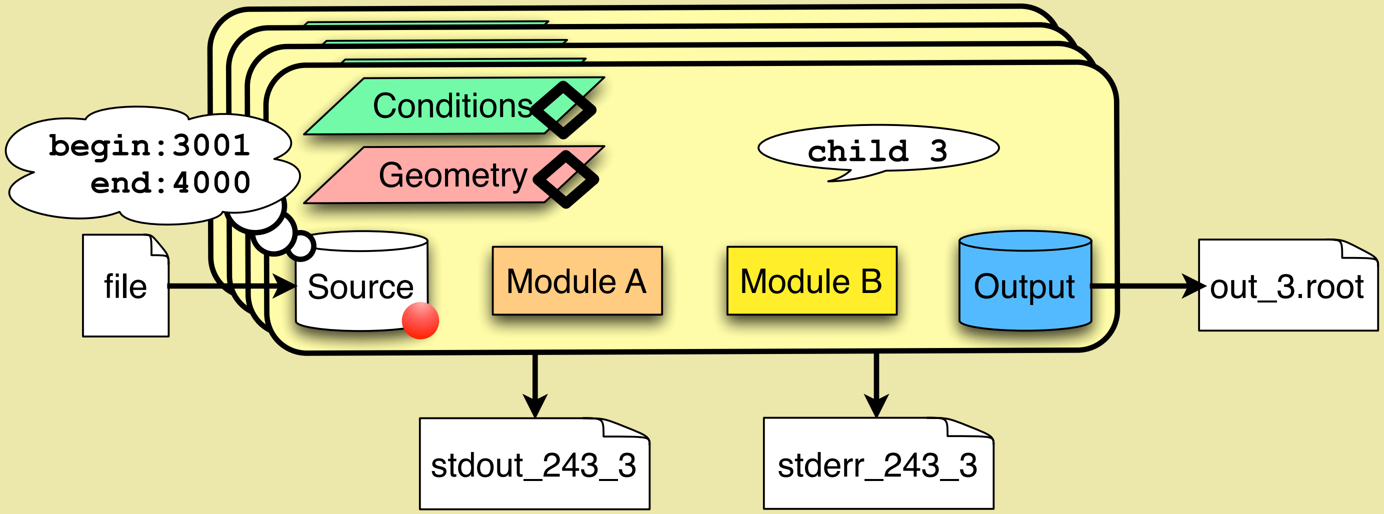
Children

Redirects stdout and stderr to own files whose names contain parent PID and child #

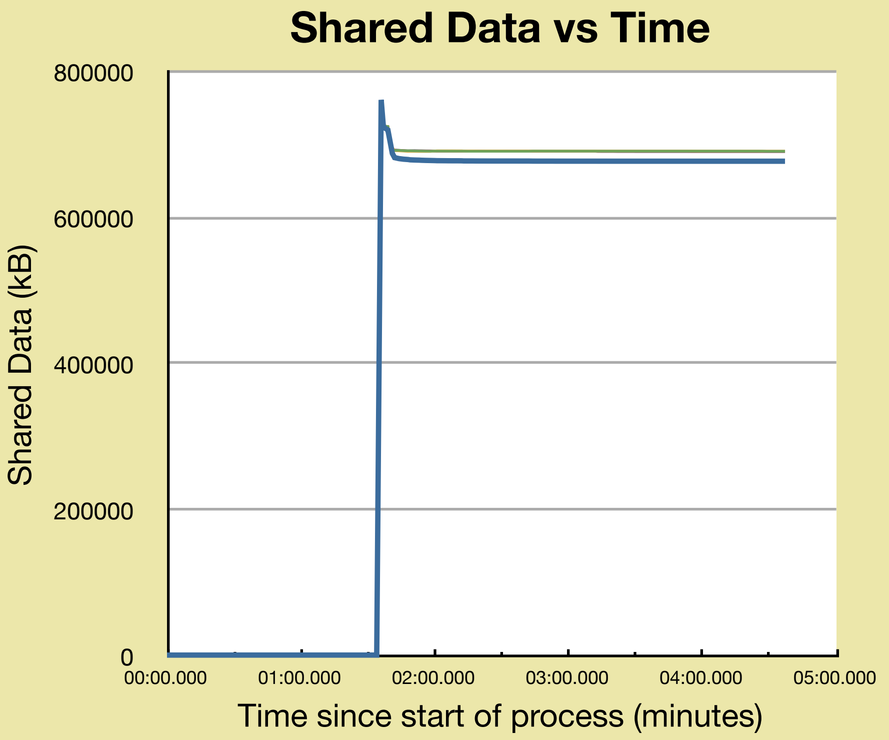
Send messages to modules saying process is child X

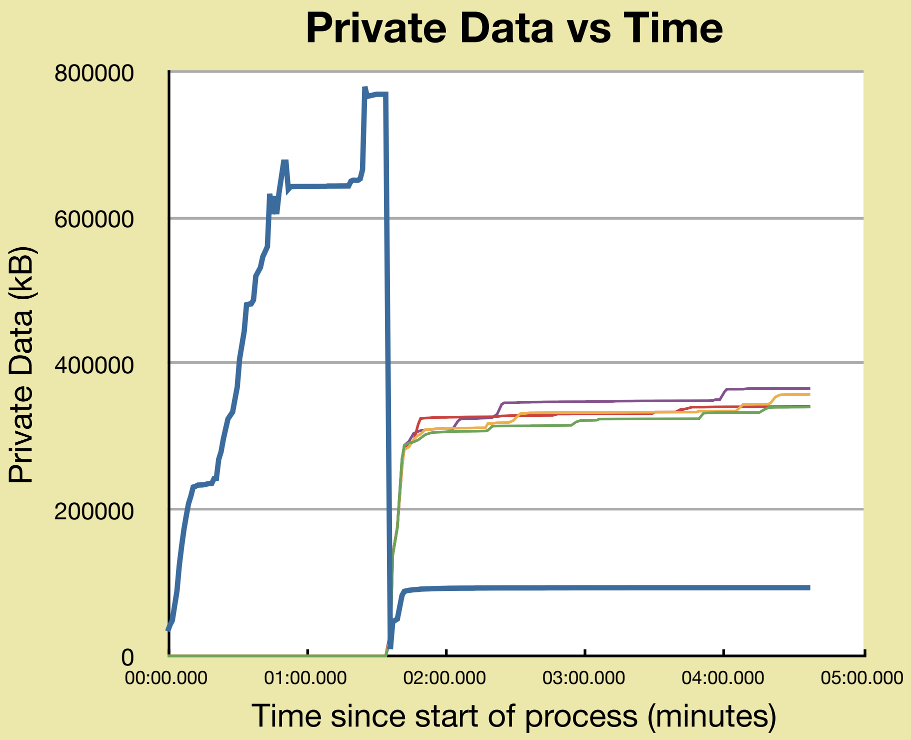
- Output modules append child # to file names
- Sources calculate their event ranges to process (no IP communication) and re-open the file

Process events in child's start/end range normally



Memory Sharing





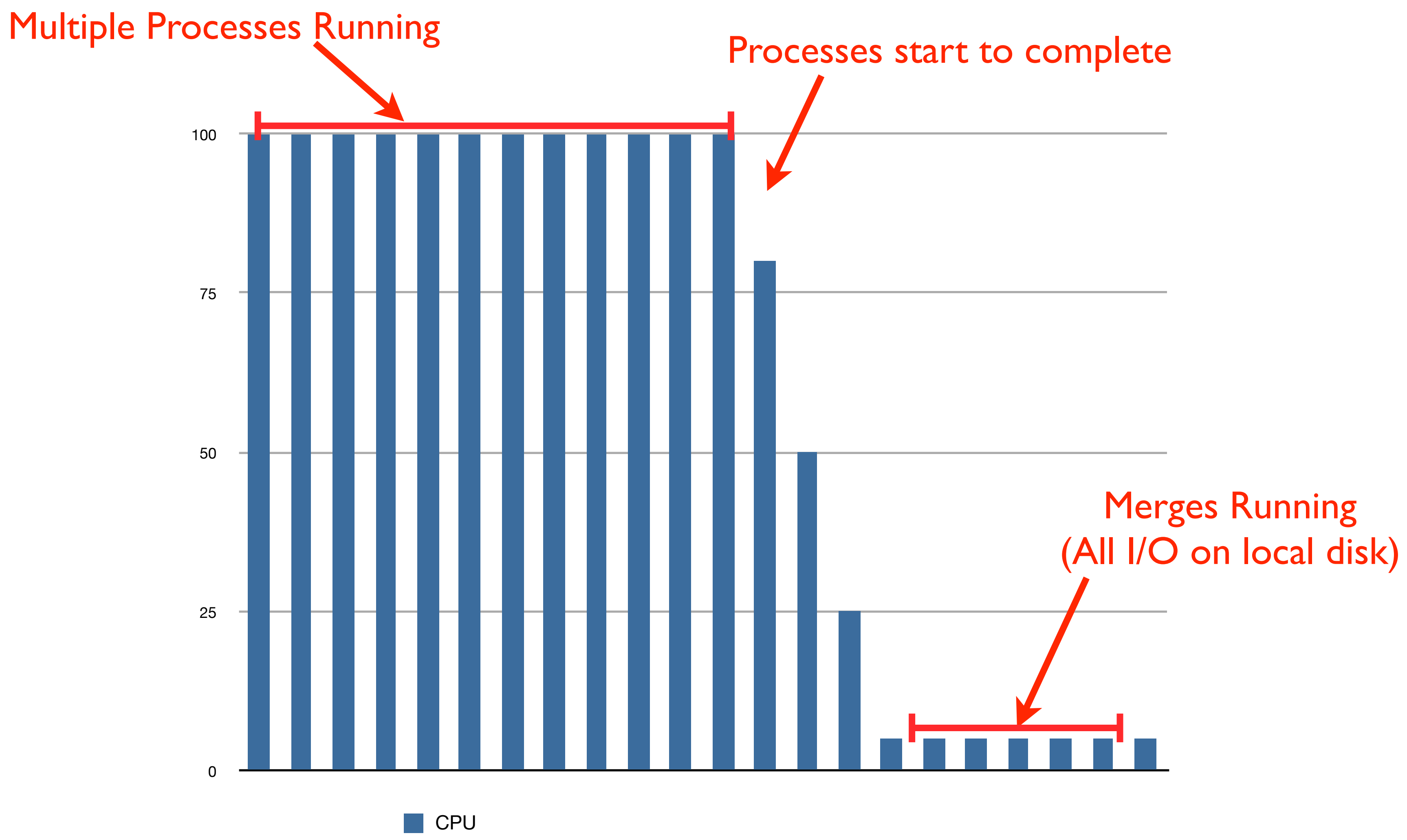
Measurements done using reconstruction with 64bit software on 4 CPU, 8 core/CPU 2GHz AMD Opteron (tm) Processor 6128

Shared memory per child: ~700MB
Private memory per child: ~375MB
Total memory used by 32 children: 13GB
Total memory used by 32 separate jobs: 34 GB

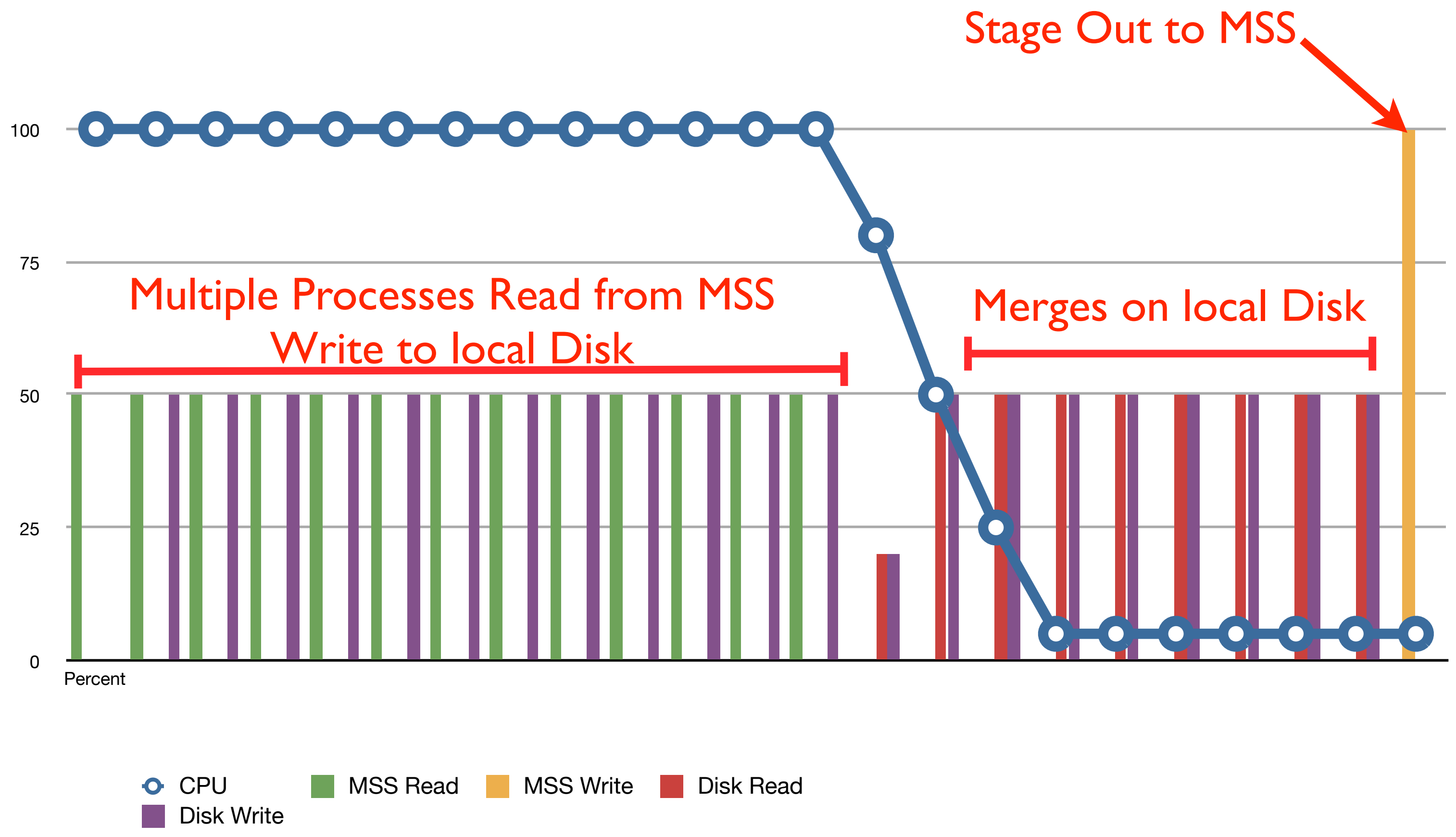
Whole Node scheduling

- All cores of a node get assigned to a multicore process
- Tier1s bill for the whole node and all the capability, so we need to make use of everything efficiently
- In principle CPU-bound workflows adapt well to multicore processing
- Multicore processing allows a decrease in the number of jobs
  - Reduced overhead in WMS
- But need to carefully evaluate the multiprocessing overhead
  - Merging, efficient use of all cores, etc

CPU Usage


Multiple Processes Running
Processes start to complete
Merges Running
(All I/O on local disk)
100
75
50
25
0
CPU

I/O



Stage Out to MSS
Multiple Processes Read from MSS
Write to local Disk
Merges on local Disk
100
75
50
25
0
Percent
CPU
MSS Read
MSS Write
Disk Read
Disk Write

Activity on multicore processing

- Whole Node Job Submission Task Force created by WLCG
  - LHC experiments, sites, CERN IT, LCG
  - Exploit multicore CPU's in a grid environment
  - Jose H/Claudio G. representing CMS
- CMSSW supports multicore processing
  - For data processing workflows (not yet for MC generation workflows)
- WMAgent supporting multicore processing
  - Data processing workflows

Whole-Node queues in CMS

- Some CMS Tier-1 sites providing already whole-node queues with limited resources (as of June 2011)
  - CMS T1 contacts asked T1 sites
  - FNAL: 25 nodes (8 cores each)
  - CNAF: few nodes shared by ATLAS
  - RAL: 4 nodes
  - PIC: preparing a queue with 1 node
  - IN2P3: in ~1 month
  - ASGC: queue existing (need to provide url)
  - KIT: ?
- Imperial volunteered to provide a whole-node queue ~after summer

WMAgent multicore testing

- Testing WMAgent scheduling of multicore jobs
- Setup (also as of June 2011)
  - WMAgent + glideinWMS factory at CERN
  - So far only FNAL queue included
  - Including CNAF and RAL. No pilots run yet there
- Running multicore data processing workflow at FNAL
  - The workflow runs fine
  - No performance monitoring yet
    - Need to merge individual job reports to get aggregated values
    - Developers working on it (trac tickets)

Resource planning

- Discussed in one of the last Friday computing meetings
- During the summer get all Tier-1s onboard with up to 5% of the resources in whole-node queues
- Have production workflows running in those nodes before adding additional resources
- By Fall increase resources to 25%?
- Transition 50% of Tier-1 resources by end of the year?
- Need to synchronized to other VOs in multi-VO sites
- Coexistence of single/multicore jobs
  - Dedicated queues? A more flexible/intelligent scheduling not to waste resources? WNs dynamically assigned to single/multicore queues?

Application to Tier3

- Analysis workflows will not use this mechanism in the foreseeable future
    - Too chaotic. Some analysis is CPU bound, some is Root-IO bound
    - No easy way to deal with user-produced histograms in forking
    - Analysis jobs can also write arbitrary files
- Really want a recommendation? Check out Tier1's systems and don't skimp on the disk