Designs/options for feedback from tools to improve performance etc. & System monitoring / utilization - integration of mapping, validation, visualization & How can more of our tooling become more interoperable?

https://tinyurl.com/stw-monitoring

<TL/DR> (outbrief)

Automatic performance optimization is hard, with or without feedback (MAQAO is an example tool that addresses it)

Efforts have been made to save time, energy - but are one-off solutions integrated into schedulers

Elastic applications could take advantage of elastic schedulers - can we find a prototype example to work with on HPC? (cloud example?) (workflow example?)

Commit to exploring integration between high level management and low level performance measurement/analysis (match up with at least 1 other tool)

Tool interoperability/outreach efforts exist - VI-HPS in Europe, for example

STW move to 2x per year? Non USA location?

Feedback

Hard problem - how to go from measured performance data to information that is actionable

Challenge - complexity of components, expert knowledge

- Intel top-down model
- AMD top-down model
- ~~IBM CPI-stack model~~
- Others?
- I/O bottlenecks - Darshan diagnostic/recommendations - what if you get bad/broken/misconfigured hardware (bad OST)
- Distinguish between facility failure vs user problem
- What is the baseline? Performance expectations?

Diagnostic distinction

Venn diagram:

- Operator understandable metrics
- User understandable metrics
- Need some crossover to interpret each other
- Alarms / tripwires?
    - Node failures result in pulling the node from node queues - opportunity to use a partially disabled node
    - Would users know what their jobs are going to exploit/require?
    - Specialized queues for jobs that are OK with less-than-ideal nodes
- Need the hooks in system software to integrate feedback from jobs

Erlangen example

- Analysis of jobs while they are running, sending metrics every X seconds, energy manager kicks in every X minutes, manages power for currently running jobs
- Outcomes are stored in an event database
- Power changes are reported to a web interface
- Race to idle is still the best for energy saving (faster is better)

How to leverage performance research

- Scheduling decisions?
- Deep tooling decisions?
- Carrots & Sticks
  - Bonus core hours for opting-in
- More flexibility for reconfiguration from users due to Cloud influence
- User feedback vs automated reconfiguration
- Smart runtime system could collect integrated information from disparate sources and utilize “/feedback” filesystem - synthesize information from sources and make decisions about control (separation of concerns)
  - Design & protocols needed
- Alternate - stack redesign with more communication, client/services

Reporting Progress

Application can report “progress” to the system

Progress metric can be used analytically to ?

- Not making progress - kill the job
- Guess at wall clock time, could extend job enough to “finish”
- Identify “imbalances” and address them
- Identify instabilities and terminate early
- Computational steering at the system level
- Ample opportunities for abuse - how to prevent
- How to encourage users to opt-in - priority queues to volunteers
- Domain knowledge is needed
- How much co-scheduling is allowed/done? Shared node access

Elasticity

- Will it ever be allowed?
- MPI Sessions
- Designed to be well balanced, static configuration
- Workflows, campaigns, AI, - all new opportunities
- Game theory needed to get users to cooperate - how to incentivize?
- Chicken & egg problem - applications aren't elastic because systems aren't elastic, because…
- Need example cases that could experiment with
    - Killer workflow & monitoring information necessary to make change

Separation of concerns

LDMS - High level information, non intrusive, actionable?

Performance tools - low level information, somewhat intrusive, actionable?

Opportunities for interaction / data sharing

- Performance tools reading from system data clients
- System daemons reading from performance tools
- How to synthesize all this information to provide an expert recommendation?
    - Drishti - recommender system for Darshan trace logs (I/O only)
    - Proton - recommendation based on GPU kernel counters
    - Stalls & stall causes

Action items!

- Identify “killer app” workflow(?) to allow for elasticity in operations
- Combining system data with performance tool data
  - LDMS/Kafka + Kokkos/Caliper/others?
  - Zerosum + LDMS/other client as data source
  - Data should/could flow both directions
  - Data reduction necessary? Statistical sampling? Summary metrics?
  - Combining in databases, not just on node
- What information is useful / actionable?
- What’s the useful life cycle for this data?
- Semi-annual meeting in non-USA location?

MAQAO examples

Take performance measurement, combine with expert knowledge, report to user in HUMAN readable form what the actionable changes are recommended AND the expected benefit.

(source code-level changes)

Important point - diagnostics are reported at source code level, and human friendly

Additional info: MAQAO overview, Code App Optimization

Tool interoperability

Common schema(s) or “discoverable” formats?

Metadata collection and usage?

ADIAK, Machinestate

VI-HPS.org – https://vi-hps.org

Every tool promises to learn about / collaborate with another tool