VI-HPS

PRACE

15th VI-HPS Tuning Workshop
Review

Marc-André Hermanns

German Research School for Simulation Sciences

Workshop review

You've been introduced to a variety of tools, and had an opportunity to try them on your own application codes
- with assistance to apply and use the tools most effectively

- Tools provide complementary capabilities
  - computational kernel & processor analyses
  - communication/synchronization analyses
  - load-balance, scheduling, scaling, …
- Tools are designed with various trade-offs
  - general-purpose versus specialized
  - platform-specific versus agnostic
  - simple/basic versus complex/powerful

Tool selection

- Which tools you use and when you use them likely to depend on situation
  - which are available on (or for) your computer system
  - which support your programming paradigms and languages
  - which are you familiar (comfortable) with using
- also depends on type of issue you have or suspect

- Awareness of (potentially) available tools can help finding the most appropriate tools

Workflow (getting started)

- First ensure that the parallel application runs correctly
  - no-one will care how quickly you can get invalid answers or produce a directory full of corefiles
  - parallel debuggers help isolate known problems
  - correctness checking tools can help identify other issues (that might not cause problems right now, but will eventually)
    - e.g., race conditions, invalid/non-compliant usage
- Generally valuable to start with an overview of execution performance
  - fraction of time spent in computation vs comm/synch vs I/O
  - which sections of the application/library code are most costly
- and how it changes with scale or different configurations
  - processes vs threads, mappings, bindings

Workflow (communication/synchronization)

- Communication/synchronization issues generally apply to every computer system (to different extents) and typically grow with the number of processes/threads
  - Weak scaling: fixed computation per thread, and perhaps fixed localities, but increasingly distributed
  - Strong scaling: constant total computation, increasingly divided amongst threads, while communication grows
  - Collective communication (particularly of type “all-to-all”) result in increasing data movement
  - Synchronizations of larger groups are increasingly costly
  - Load-balancing becomes increasingly challenging, and imbalances increasingly expensive
    - generally manifests as waiting time at following collective ops

Workflow (wasted waiting time)

- Waiting times are difficult to determine in basic profiles
  - Part of the time each process/thread spends in comm/synch operations may be wasted waiting time
  - Need to correlate event times between processes/threads
    - *Periscope* uses augmented messages to transfer timestamps and additional on-line analysis processes
    - Post-mortem event trace analysis avoids interference and provides a complete history
    - *Scalasca* automates trace analysis and ensures waiting times are completely quantified
    - *Vampir* allows interactive exploration and detailed examination of reasons for inefficiencies

Workflow (core computation)

- Effective computation within processors/cores is also vital
  - Optimized libraries may already be available
  - Optimizing compilers can also do a lot
    - provided the code is clearly written and not too complex
    - appropriate directives and other hints can also help
  - *MAQAO* can help analyse and optimize instructions
  - Processor hardware counters can also provide insight
    - although hardware-specific interpretation required
  - Tools available from processor and system vendors help navigate and interpret processor-specific performance issues

Review of featured VI-HPS tools

- **MAQAO**
  - low-level code optimization for x86-64 architecture
- **Score-P**
  - community-developed instrumenter & measurement libraries for parallel profiling and event tracing
- **CUBE** & **ParaProf/PerfExplorer**
  - interactive parallel profile analyses
- **Vampir**
  - interactive event-trace visualizations and analyses
- **Scalasca**
  - automated event-trace analysis
- **TAU/PDT**
  - comprehensive performance system