Common performance issues in threaded code

David Levinthal Apr 1

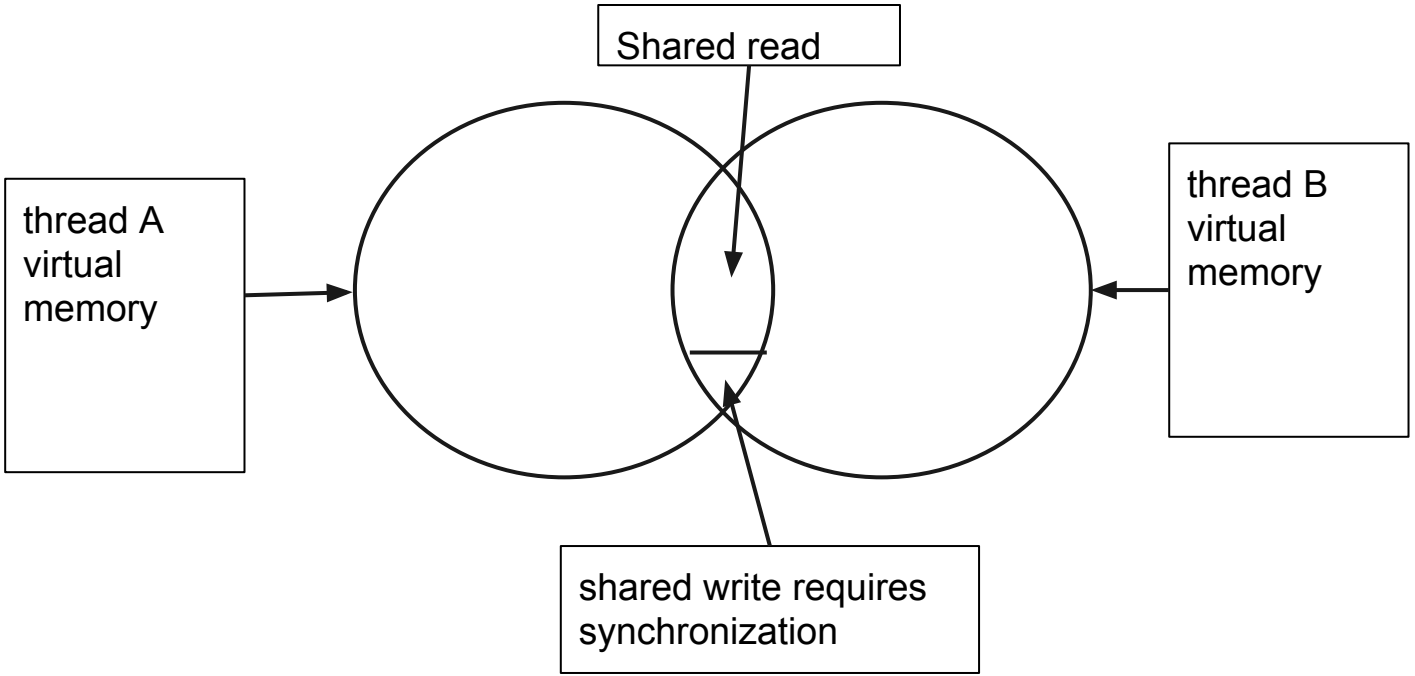
Common performance issues associated with threaded code

Lots of industry and academic experience with performance and threading

A few of the more common ones and their detection

- **Synchronization**
  - **false sharing**
- **Concurrency**
  - **load balancing**
  - **interupt thrashing**
- **Thread library overhead**
  - **OpenMP issue**

Synchronization



Controlling race conditions

Lock gated access is the normal method to avoid race conditions with modifying shared data

- **Avoid split line locks at all cost**
- **Balance lock “granularity”**
  - **HLE/TSX/HW supported transactional memory can assist with common locks controlling non overlapping data**
    - **HSW/Intel and IBM**
- **Thread can go idle when unable to acquire lock**

Existing HW event tools

VTune/Amplifier has time line displays/thread and critical path analysis

perf report has a recently added address analysis (I have not used this)

PTU (EOL'd but may still be available) has very sophisticated address analysis

False sharing

Loads/stores to cachelines with fields that get modified by other threads

    But with no overlap

Store cause line to go to exclusive state and then go to modified (M mesi state)

Subsequent load to such a line from another thread causes a HITM (hit modified)

Identifying false sharing

Address analysis

- **Use PEBS HITM and store events**
  - **mem_load_uops_retired:llc_miss_remote_hitm**
  - **mem_uops_retired:any_store**
- **HSW captures virtual address for all memory operation**
- **Compare addresses and sizes**
- **Identify non overlapping loads and stores from different threads**
- **See the old PTU tool for an example**
- **PERF recently added such a feature based on the load latency event**
- **And yes..one of these days I wll add this to Gooda :-)**

Concurrency issues

- Load balancing in data decomposition
  - Threads going idle may not be the problem
  - Busy threads may have too much work
- Broadcast event interupt on uncore clock
  - generate PMI on every HW thread
  - random sampling of concurrent execution

Context thrashing

- Exclusive locks can cause context switches to the idle state (1 thread /HT)
- Need to check context switch rate and initial/final thread
- Needs kernel instrumentation
- note: here idle thread denotes a problem

Summary

Large variety of performance bottlenecks

A problem signature may be good or bad for the specific thread (ex: idle core)

Do not assume threading is more efficient than process parallel