(1) Communication graphs
(2) Tools that offload to GPUs

Discussion during the tools meeting

Ask for edit permission by clicking

http://tinyurl.com/Solitude18GaneshBreakout

Communication Graphs (summary of discussions)

Participants: Phil Roth, Kevin Huck, Felix Wolf, David P, Ganesh G;

Ask for edit/view permission: http://tinyurl.com/Solitude18GaneshBreakout

- Generalize notion of communication matrices and graphs
  - Include things like ranks, communicators, logical/physical topologies -- even cabinets etc
- Find not only when current pattern sustains -- record transitions to new / non patterns
  - Sometimes it may degenerate to a known hairball -- e.g. embedded FFT pattern
- Train machine-learning models to recognize patterns
  - Recognize primary pattern at current level of detail
  - Do “sky subtraction” and then go after patterns at the next level of detail
- Training machine-learning models needs labeled data
  - Parametrically generate several communication models to serve as labeled data
  - For instance, point-to-point comm can be thrown in; introduce controlled randomness
- Recording with edge-weights can serve the needs of perf (comm volume)
- Correctness (relative debugging) can find what changed in comm graph
- Elastic MPI : new challenges that would be good to discuss (Michael Gerndt)
  - Patterns may change

Debugging tools that offload to GPUs (disc. summ.)

Participants: John M-C, Ben Woodward, Ganesh G, a couple of beers

Ask for edit/view permission: tinyurl.com/CommunicationGraphsSolitudeWorkshop18

- Discussed expedient path to tracking GPU synchronization
- Ben brought up PTX-based instrumentation as a way to proceed
- Decided that PTX-based instrumentation and barrier inference may be a smart way to get some things done
- Ganesh has some concerns this will do for the long-haul (see next slide)
- John sent some literature to get barrier inference done
  - DOI=http://dx.doi.org/10.1145/209936.209952
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.8519&rep=rep1&type=pdf
  - DOI=http://dx.doi.org/10.1007/978-3-540-69330-7_13
  - http://titanium.cs.berkeley.edu/papers/kamil-yelick-lcpc05.pdf
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.1283&rep=rep1&type=pdf

Why it may not work for the long-haul

- barrier inference can help but alone does not cut it
- - inter-block races
- - races in codes that combine barriers and GPU atomics
- - races avoided by fences in various scopes
- - "porting races" (conditionals unordered in evaluation)
- - usage of the right warp reconvergence primitives OK?
- * infer behavior around "shuffles" (any)?
- * the sync primitive used in Alex Aiken's chemistry codes
- (like a named barrier)
- * New warp-level primitives
- __activemask and __syncwarp
- * Opportunistic warp-sync programming
- - implicit warp-sync programming is dangerous
  - Detect such bugs too
- Existing GPU verification tools (partial list)
  - GKLEE (PPoPP'12, SC'15), GPUVerify (Donaldson), CURD (Devietti)

DISCUSSION ROUGH NOTES

Discussions

- FW: Shared mem accesses (patterns in)
- PR: Demand not just for MPI but also comm across other APIs (accelerators)
- KH: Data exchange to (between) libraries
    - ADIOS, Data Spaces, SST
- DP: Interested in using it for applications where ranks have diff characteristics
    - KH, PR: Coordinates for ranks (cabinet, 2D/3D pat),,Hypercube, GPU offload in-between
    - PR,KH: Patterns around diagonal; Distill things like nearest-neighbor exchange
    - DP: Found patterns till m,n; failed patterns at p,q; Could it be sub-communicators?
    - KH,PR: need to track comm creation
  - Languages for pattern description
    - FW: # comm partners, amt of data exchanged. Mine locality info.
    - PR: Clustering procs based on metrics?
  - PR: for debugging: ScalaTrace: Scalable compression and replay of communication traces for high-performance computing (Muller's direction of work)
  - FW: have done it for task graphs (Umps framework?). Can get metrics (work/depth)
  - DP: rank-based semantics would be good to mine.
    - Relative values of communication volume, bytes exchanged etc.
  - GG: Concept lattices may be a good way to summarize rank-specific features. Here is a use of CLs in the perf space: Structural Clustering: A New Approach to Support Performance Analysis at Scale
  -

Discussions

- PR: Can we include more info like taint info.
- KH: MPI with threads
- PR: Karen has done work on comparing results from run1 to run0 in terms of perf
- KH: Solver “nondeterminism” in terms of how convergence happens. FFT suddenly engages in a different pattern.
    - KH: May want to ignore “hairballs” that pop up in the middle
    - PR: mine phases and then say what’s of interest (or not)
    - KH: capture data wrt communicators gives us handle on ignoring things efficiently
    - KH: Patterns may be generated perhaps using ML-techniques
    - DP: Proving one is wrong wrt pattern mining within small instances may be efficient
    - PR, KH: Greedy attribution (automation) may be error-prone, but ML may help pick out those “human recognizable patterns”. This is after “sky subtraction” is done.
    - FW: Need enough training data.
        - One can focus on pt-to-pt and then focus on collective calls
        - DP: some info on geometry is available. Logical/Physical layout
        - GG: contain pattern-space to what’s feasible
        - PR: maybe fold in FW’s shared memory info
        - Graph-generation for benchmarking graph-analysis tools/algos is in this IPDPS’18 paper
            - Communication-free Massively Distributed Graph Generation
        -

Discussions

- DP: graph-generation may be useful in generating training data for ML tools (tagged/labeled data)
- KH: We are interested in some principal patterns; can we parametrically fill in noisy (biased) nearest-neighbor?
- PR: LAMMS situation where generating test cases..
- KH,PR: Data volume and calls.
- PR, FW: Scalasca - late-sender [https://dl.acm.org/citation.cfm?doid=2974644.2934661]
- KH: logical/actual time diff is where problems are
- PW,GG: This is how patterns were used in “industrial-scale cache coherence verification”
    - http://www.cs.cmu.edu/~tmurali/pubs/fmcad09.pdf
    - DP: might we want to put something through multiple learning sequences for patterns?
    - KH: probabilistic match for what pattern did we end up matching
    - PR, GG:

Pre-discussion slides (Blame-shifting to Ganesh)

Community interest in debugging

- DOE report : http://tinyurl.com/DOE-HPC-Correctness-2017-pdf

- HPCWire article

  http://tinyurl.com/DOE-HPCWire-Correctness-2017-pdf

Gist

- No way to diagnose a large-scale crash/hang other than
  - Attach tools such as STAT
    - Info available at that point is not voluminous
- Approach desirable
  - Maintain more information even during a healthy-looking run
    - When crash-hang occurs, we can compare against healthy-run events from a prior successful run

Gist

- What to collect
  - User specifies salient events
    - Collected events compressed and stored
  - When we decompress what to do
    - Decompress and on-the-fly build features
- This way, the collected info can help diagnose crash
- Differential debugging (what went wrong from past working version to now)

Comm graphs

- While doing decompress and on-the-fly build features
  - Suppress symmetries
  - Highlight outliers
- Symmetries are mined through
  - Comm graphs
  - Loop detection
  - Other ideas
- What’s good for debugging is a good starting point for correctness

Sales

- People will use debugging tools
  - Correctness tools coming attached is a good idea
- Debugging needs happens-before
  - This serves as critical-path info for perf tools
- Synergy between perf (elephant) and debugging (mouse) is greatly desirable