Challenges for Developing & Supporting HPC Applications

William Gropp
wgropp.cs.illinois.edu
HPC Applications are about solving problems
Programming is only one small(ish) part that includes
- Data management (input, output, visualization, analytics, ...)
- Performance and correctness debugging
- Integration with workflows
- And yes, dealing with limitations in programming systems and their implementations, particularly the growing diversity in architectures
Speakers

• Torsten Hoefler, ETH Zürich
  • Automatic compiler-driven GPU acceleration with Polly-ACC

• Jeffrey Hollingsworth, University of Maryland
  • Bugs and Speed in HPC Applications: Past, Present, and Future

• Michela Taufer, University of Tennessee, Knoxville
  • Modeling the Next-Generation High Performance Schedulers
HPC Application Lifecycle (Partial)

- **Design**
  - Choice of mathematical models
  - Choice of algorithms

- **Implementation**
  - Choice of programming approaches
  - Choice of Libraries and the use of code analysis tools

- **Testing**
  - Correctness
  - Performance (and do you know what the achievable performance is?)

- **Science workflow**
  - Creating input data and analyzing output data - includes mesh generation (e.g., CFD) or data partitioning (e.g., bioinformatics)
  - Run ensembles for uncertainty quantification, parameter sweeps, nonlinear optimization, ...

- **Repeat each step in all combinations...**
Real Challenges in Programming

• For HPC, we are looking for high performance
  • FLOPS and Memory Bandwidth ("roofline" https://dl.acm.org/citation.cfm?id=1498785 )
  • FLOPS and Memory Bandwidth and Latency (Execution-Cache-Memory (ECM) model https://link.springer.com/chapter/10.1007%2F978-3-642-14390-8_64 )
  • FLOPS and Memory Bandwidth and Instruction Rate ("Achieving high sustained performance in an unstructured mesh CFD application" https://dl.acm.org/citation.cfm?id=331600 , 1999)

• Node performance is often key (in 1999 result above – 7x performance improvement from memory locality on the node)

• In distributed memory programming, the challenge is managing the distributed data structures and the operations upon them
  • It would be great if a programming language provided your data structure (and some are close) but the reality is that most apps have specific needs
Managing Code Transformations

• Many tools exist (some you’ll hear about today)
• Need a way to
  • Separate additional abstractions (e.g., loop count is small) vs. proscriptive requirements (e.g., unroll loop by 3)
  • Invoke multiple tools
    • Transformation generators, autotuners, ...
  • Remember good (and bad!) choices of parameters, transformations, etc. by system/input/characterization
  • Provide ways to confirm transformations preserve correctness
What is Correctness?

• How do we know that the performance portable code is correct?
  • Or even if it will compute the same result as the original code
  • And what is “the same result”?

• It is not enough to prove that any code transformations are correct
  • MPICH used to test whether the compiler returned the same result in a and c for these two statements:
    • \( a = \text{joe->array[OFF+b+1]} \);
    • \( c = \text{joe->array[OFF+1+b]} \);
  • Because one major vendor compiler got this wrong.

• And you still need to prove that the hardware implements all of the operations correctly
  • And vectorization is already likely to produce results that are not bitwise identical to the non-vector version (which might depend on how data is aligned at runtime)

• Question: How do you test that the performance portable code is computing what is intended?

• Proving code transformations correct is necessary but not sufficient
Illinois Coding Environment (ICE)

- One pragmatic approach
- Assumptions
  - Fast code requires some expert intervention
  - Can’t all be done at compile time
  - Original code (in standard language) is maintained as reference (*Golden Copy*)
  - Can add information about computation to code
- Center for Exascale Simulation of Plasma-Coupled Combustion
  - [http://xpacc.illinois.edu](http://xpacc.illinois.edu)

- Approach
  - Annotations provide additional descriptive information
    - Block name, expected loop sizes, etc.
  - Source-to-source transformations used to create code for compiler
    - Exploit tool ecosystem – interface to existing tools
    - Original “Golden Copy” used for development, correctness checks
  - Database used to manage platform-specific versions; detect changes that invalidate transformed versions
Work of Thiago Teixeira and David Padua

---

#pragma @ICE loop=stencil
for(i = 1; i < x-1; i++) {
    for(j = 1; j < y-1; j++) {
        for(k = 1; k < z-1; k++) {
            B[i][j][k] = C0 * A[i][j][k] + C1 * (A[i+1][j][k] + A[i-1][j][k] +
                                A[i][j+1][k] + A[i][j-1][k] +
                                A[i][j][k+1] + A[i][j][k-1]);
        }
    }
}
#pragma @ICE endloop

#Built command before compilation
prebuildcmd:

#Compilation command before tests
buildcmd:
    make realclean; make CC={compiler} COPT={params}
buildoptions:
gcc:
    params: {'-O':'default': 3,'min': 0,'max': 3}
icc:
    params: {'-O':'default': 3,'min': 0,'max': 3}

#Command call for each test
runcmd: ./sten3d 1024 20

tuning: on

stencil:
    rose_uuiuc:
        - stripmine+:
            loop: 4
            factor: 16..1024
            type: poweroftwo
        - stripmine+:
            loop: 3
            factor: 16..1024
            type: poweroftwo
        - stripmine+:
            loop: 2
            factor: 16..1024
            type: poweroftwo
        - interchange+:
            order: 0,1,3,5,2,4,6

...
Performance Results

- 3-D Stencil
  - 11,664 variants
  - Max 12.6 sec
  - Min 3.68 sec
  - Speedup over simple code
    - icc: 1.12x
    - gcc: 1.21x
Summary

• HPC Applications require many kinds of support over their lifetime, especially beyond programming
• Many tools and approaches exist
  • A challenge is to make these tools work together
• (though I have not discussed this) HPC and “Big Data” environments share problems and solutions
  • MPI and scalable algorithms for collective operations from HPC used in ML
  • Data systems and tools from big data offer better capabilities and user productivity for HPC
  • Only a start here. Both sides have much to learn and to offer