Exploring the Concept of “Motifs”

S.D. Hammond, M.J. Levenhagen, K.S. Hemmert, B.J. Moore and A.F. Rodrigues
Scalable Computer Architectures, Sandia National Laboratories
Feedback for ModSim Community

- “We like the idea of application models”
  - Lots of positive comments about how these can be useful
  - Surprising number of application developers knew of models of application performance
  - Many saw it as a potential help for getting to Exascale
  - Become popular during recent DOE procurements

- Industry interaction (particularly DesignForward)
  - Traces are too brittle, don’t scale and aren’t very insightful
  - Want something truly flexible and easy to reason about
  - Want something simple
  - Want the DOE to lead more of the application work
    - Make these available as part of procurements
    - “Reference” implementation of models from the people who own the applications
Thoughts

- Traces of application communications have really been a misfire for the community
  - Too inflexible to scale to large processor counts
  - Lack sufficient detail to relate message sizes *etc* to scale, problem size, processor counts *etc*
  - Too large in GBs-TBs to be practically useful
  - Too brittle to allow exploration of “what ifs”

- We need to do something **beyond** traces
Conventional Approach

- Most conventional approaches are to
  - Take your application source code
  - Study it for kernels and communication/annotate it *etc*
  - Map this on simulation or an analytical model operation by operation
  - Analyze behavior through the model

- This is really top down, usually done by the “performance modeler” or some expert
  - Keep hearing at ModSim that there are not enough modelers
  - Complex when you have anything that isn’t a mini-app
  - Time consuming
  - Accurate?
Using “Motifs” as a building block

- We see “motifs” as describing basic communication patterns that are replicated over many applications
  - Frequent repetition of patterns across application domain
  - Usually differentiated only by parameterization

- Want to model a workflow as a combination of these motif building blocks
  - Easily allow developers to explore change in the combination

- Including but not limited to:
  - Two sided MPI
  - One sided MPI and OpenSHMEM
  - I/O patterns
Where are we?

- Reference implementation implemented in SST
  - Halo2D/3D patterns, Sweeping, FFTs, some solver schemes
  - Prototypes for AMR, AMG and graph analytic applications

- Currently mapping procurement apps and mini-apps onto these blocks

- Demonstrated scaling to 8M virtual MPI ranks in recent runs (in a small number (16) physical nodes and realistic simulation times)

- Accuracy of some initial models is >85-90% without attempting to tune models (out of the box experience)
Motif Patterns

- Not just an “SST” thing
  - Part of a (ModSim) community effort to build a library of commonly used patterns
  - Part of the codesign space to explore what-ifs around these for hardware and applications
  - Compatible with PALM, ASPEN, analytic models, etc etc

- Truly “Open” for business
  - Open source implementation (BSD so good for vendors)
  - Open as a process - open library for everyone to use
  - Open for all vendors to use as part of their research and procurement
Challenges for the Community

- **Flexible representation of a workflow**
  - Something composable from many developers
  - Includes analytics, visualization, I/O and traditional computation

- **Sufficiently flexible to be studied as “what if...”**
  - Needs easily replaceable sections or pieces of a model
  - Without needing to recode and re-extract kernels etc

- **Easy for application developers to use and enhance**
  - Not the traditional byzantine mess of simulators and models
  - Easy to experiment without needing the expert in the loop

- **Scalable within a single definition**
  - Number of PEs is a parameter, not fixed by a single run/set of runs