Modelling the Performance of a Shallow Water Code

Computational Modelling Services
NATIONAL CENTRE FOR ATMOSPHERIC SCIENCE
NATURAL ENVIRONMENT RESEARCH COUNCIL

Annette Osprey[1], Graham Riley[2], Bryan Lawrence[1] and Muniyappa Manjunathaiah[3]

a.osprey@reading.ac.uk

1 Department of Meteorology, University of Reading, 2 School of Computer Science, University of Manchester
3 Department of Computer Science, University of Reading

1. Performance modelling

A performance model is an equation for the time to run an application on a given machine, based on software and hardware characteristics.

Performance models can be used to:

- *understand* current behaviour;
- *make predictions* about future behaviour; and
- *inform* design choices when developing or updating code, for example in Gung-Ho!

2. Shallow water model

The application studied here is a simple program *shallow*, based on the NCAR shallow water model: http://www.cisl.ucar.edu/docs/hpc_modeling/

Shallow mimics key parts of more complex climate model applications such as the UK Met Office Unified Model, particularly:

- time-step iterations;
- loop-based calculations to update array values based on other arrays;
- exchanges of boundary data between cores.

3. A performance model of shallow

Our performance model takes the form:

$$T_{total} = T_{comp} + T_{comm}\ ,$$

where:

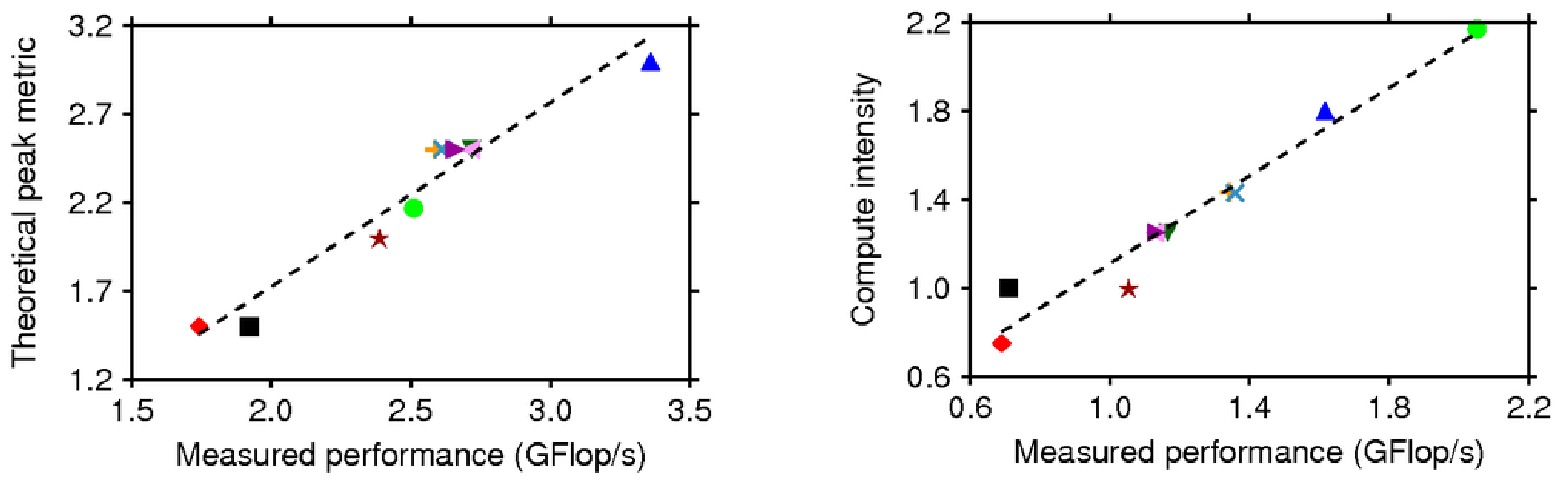
- T_{comp} is the time spent in computational loops updating array values, and
- T_{comm} is the time spent performing MPI data exchanges between cores.

The times for each part are derived from code analysis and benchmarking.

4. Computations
The single core loop performance

Shallow performs ten array update loops each timestep. The time to complete each one depends on the operations to be performed (adds, multiples, memory loads and stores), the location of data in the cache or memory system, and factors such as the compiler and processor hardware. It is possible to estimate the *relative* performance of each loop from a simple analysis of the source code and compiler transformations.

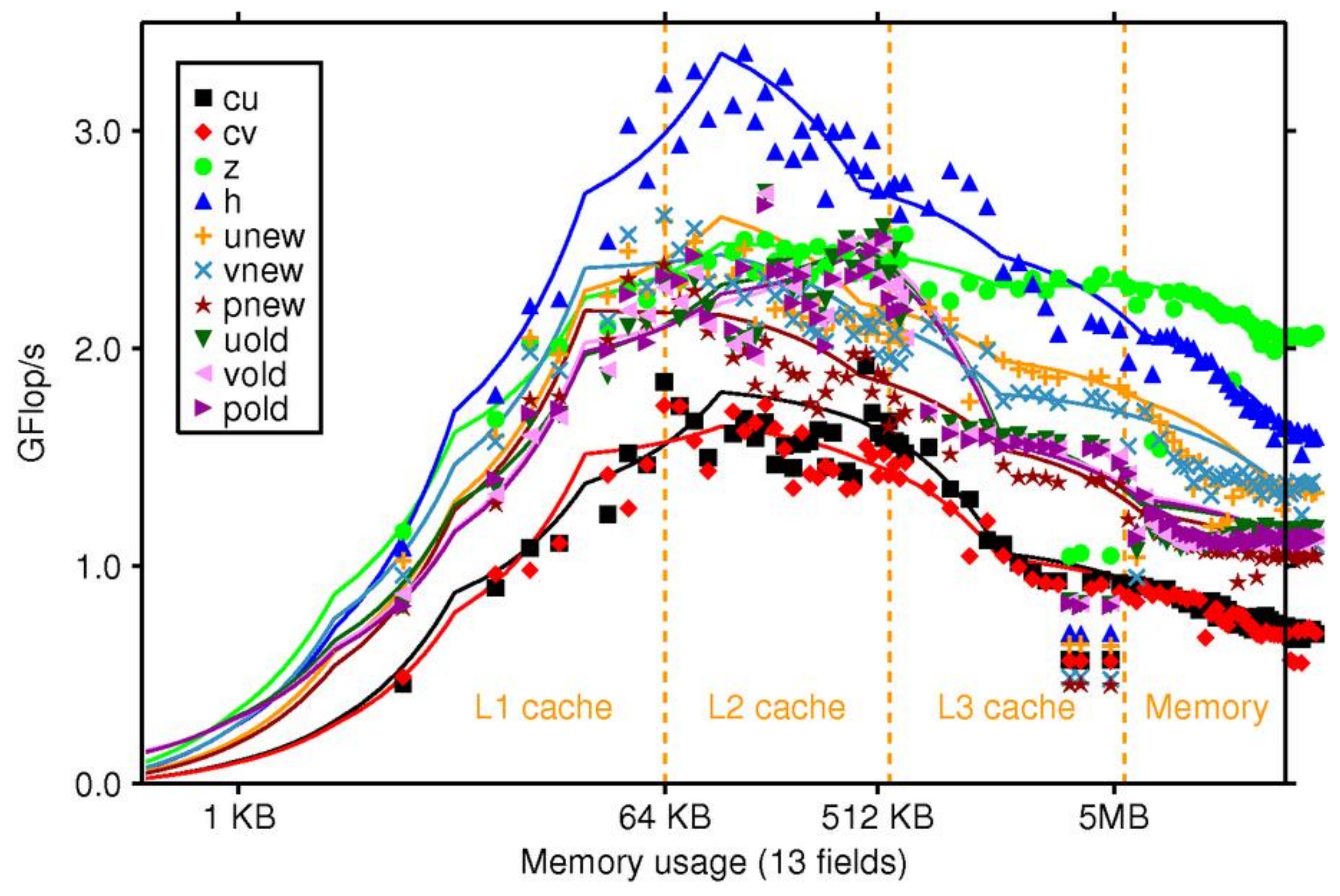
Theoretical estimates versus measured performance.


a) Peak performance (optimal problem size for cache).

b) Performance in memory (very large problem size).

Experiments were performed on HECToR Phase2b with the PGI compiler. The Phase2b was a Cray XE6 comprising Magny-Cours processors with 16 KB of L1 cache, 512 KB ofL2 and 5 MB of L3.

To derive *actual* performance values requires some level of machine benchmarking. In this case the entire loops are measured for a series of exponentially increasing problem sizes. The predicted performance for any problem size is then based on a linear interpolation from this data.



5. Communications
The performance of boundary exchanges

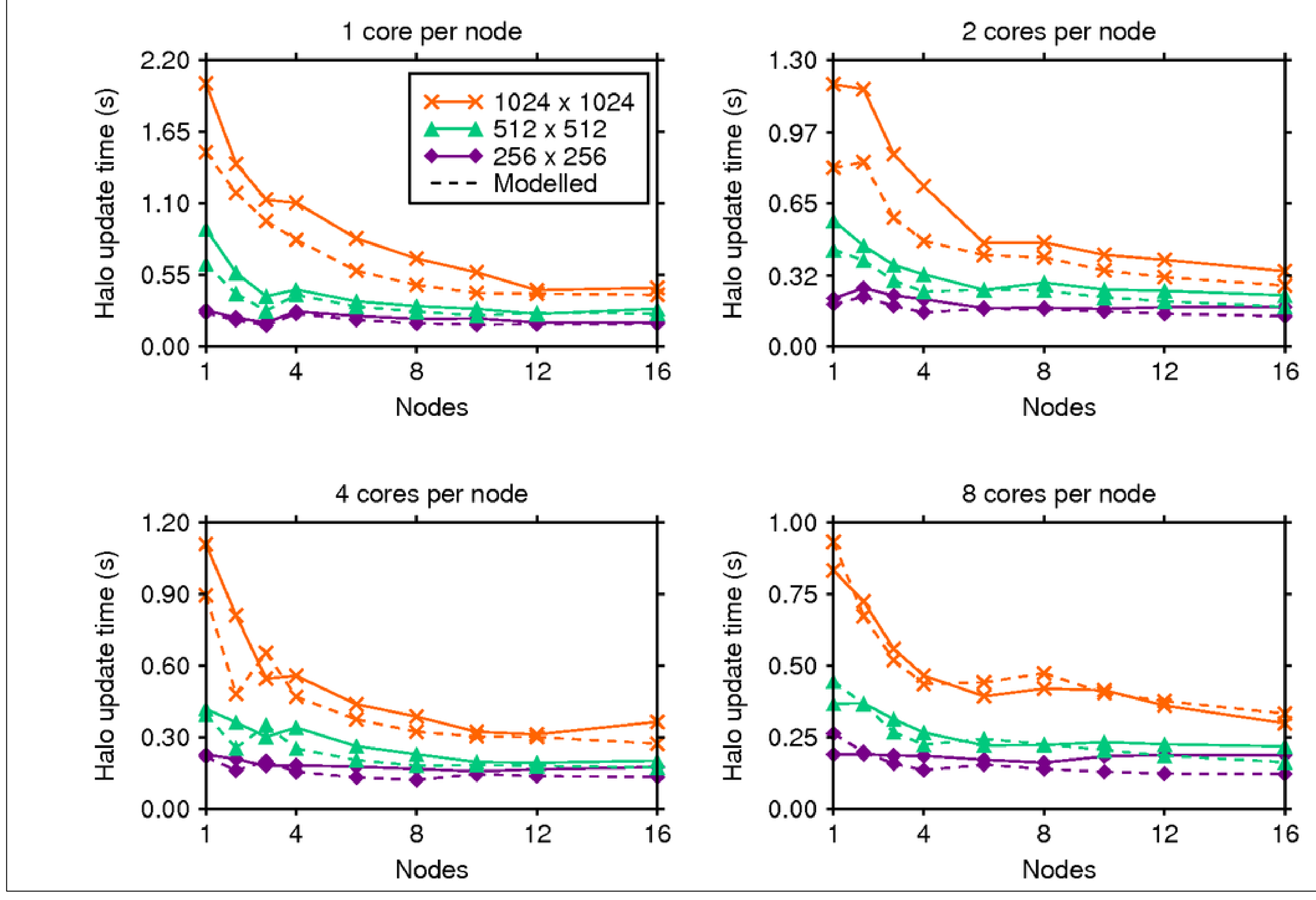
Communication experiments were performed on the HECToR Phase 3, a Cray XE6 with Gemini interconnect and Interlagos processors. Nodes are comprised of 4 dies each containing 8 integer cores. So far tests have only been run on an empty test and development system identical to the main machine to avoid network interference from other jobs.

At each timestep seven fields perform boundary exchanges. These are done in two directions only – with one row and one column being sent and received each time. The major factors affecting the boundary update time are:

- *The MPI transfer time*: Based on an MPI_Sendrecv operation, this depends on whether the transfer takes place between nodes, or within a node, and the number of concurrent transfers taking place along the link.
- *The data access time*: This depends on the data location in cache or memory, and whether the access is contiguous or non-contiguous.

A model was developed to take a given domain decomposition and work out the halo sizes, where the transfers take place and the data access pattern. Times were then taken from a set of benchmark data.

Predicted time to complete halo exchanges (dashed lines) and measured times (solid lines) for different problem sizes, cores per node and total number of cores.



6. Next steps
Finishing the performance model

- Include multi-core loop performance with shared cache effects.
- Update computation model to Phase 3 with Interlagos processor.
- Put computation and communication models together.

Future work

- Communications benchmark on the main HECToR system for some measure of run time variability.
- Evaluate model process on a different machine.