Argonne NATIONAL LABORATORY

Panel: System Software Issues for the Future

William Gropp
Mathematics and Computer Science
www.mcs.anl.gov/~gropp

THE UNIVERSITY OF CHICAGO
Office of Science U.S. DEPARTMENT OF ENERGY
A U.S. Department of Energy laboratory managed by The University of Chicago

Panelists

- Bill Gropp, panel chair (Argonne)
- Satoshi Matsuoka (Tokyo Tech)
- Alok Choudhary (Northwestern)
- Henry Tufo (NCAR)
- Pete Beckman (Argonne)
- Todd Inglett (IBM)

Questions for Panel

1. From the perspective of system software, what are the biggest challenges facing users of BG?

   (Users can be system administrators or computational scientists. System software is anything that the user doesn't write.)
2. How can we quantify that challenge?
3. What can be done in the next year? In the next two years? Which of these are primarily software and which are strongly affected by hardware?
4. Are there opportunities for collaboration in solving some of these problems?

My Answers

1. Biggest challenge: Sustained performance
2. Quantification:
    1. Per-node performance, particularly with respect to systems of similar capability, and “scaled” processors (define a “processor” as the number of CPUs that are needed for 1 GFLOP of achieved performance)
    2. Scalability (once per-node performance is good)
3. What can be done: Must be software; better tools to transform code
4. Collaborations? Yes!
    1. Shared experiences in performance artifacts (Wiki?)
    2. Library of code transformation templates

Sustained Performance

- Single node
  - Using “double hummer”
  - Managing memory hierarchy
- Parallelism
  - Managing topology
  - Detecting performance deficiency
  - Making efficient use of BG features (such as concurrent communication on each link)
- Interrogation
  - Did you get what you expected?
    - *Use of double hummer*
    - *Cache efficiency*
    - *Did system apply specified topology?*

Annotations example: STREAM triad.c for BG/L

```
void triad(double *a, double *b, d
{
  /* --Disjoint;;var:a,bc --*/
  int i;
  double ss = 1.2;
  /* --Align;;var:a,b,c;; */
  for (i=0; i<n; i++)
    a[i] = b[i] + ss*c[i];
 /* --end Align */
}
```

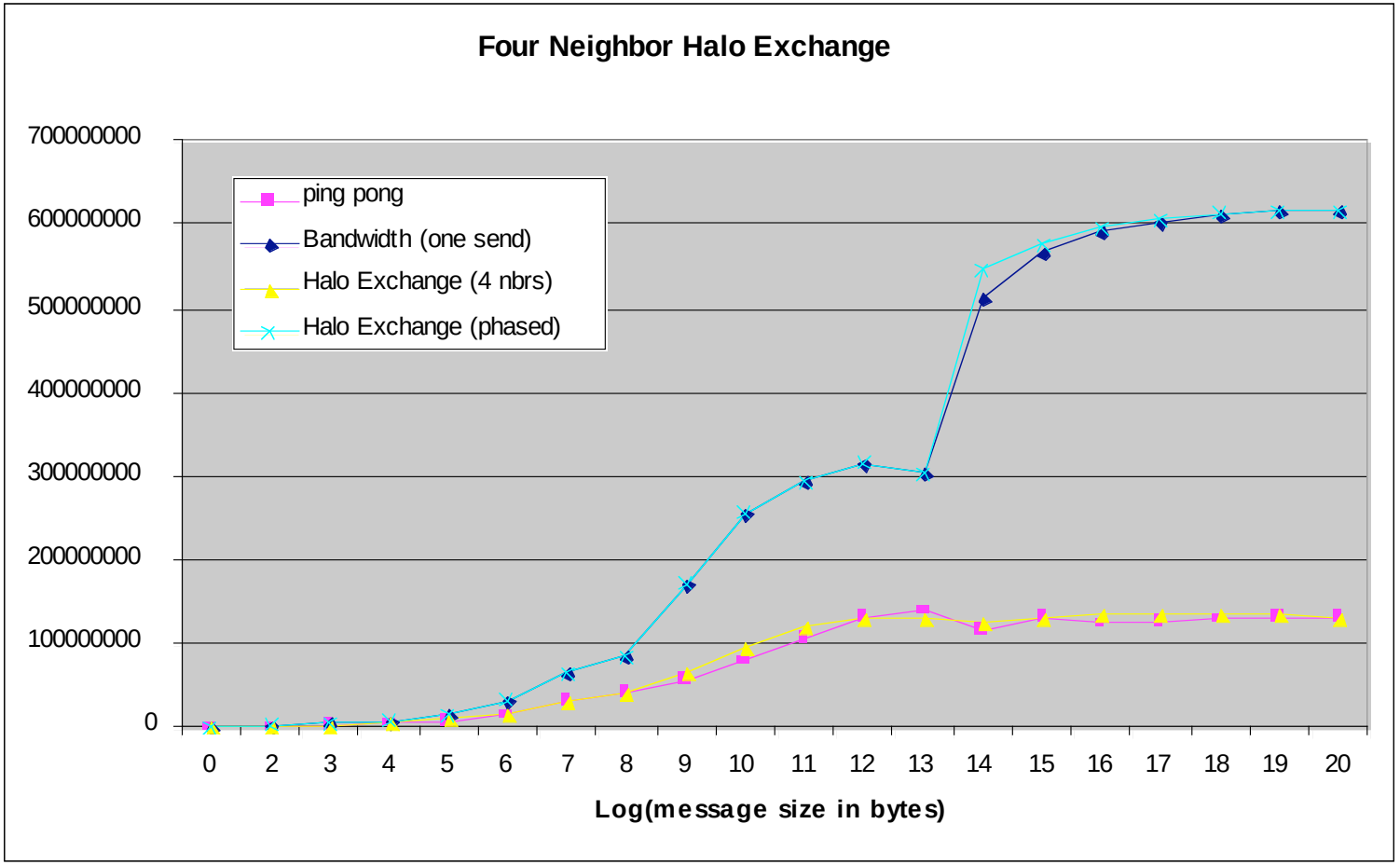
```
void triad(double *a, double *b, double *c, int n)
{
#pragma disjoint (*c,*a,*b)
  int i;
  double ss = 1.2;
  /* --Align;;var:a,b,c;; */
 if ( ((int)(a) | (int)(b) | (int)(c)) & 0xf == 0) {
   __alignx(16,a);
   __alignx(16,b);
   __alignx(16,c);
  for (i=0;i<n;i++) {
    a[i] = b[i] + ss*c[i];
  }
  }
  else {
    for (i=0;i<n;i++) {
        a[i]=b[i] + ss*c[i];
  }
   /* --end Align */
}
```

Resulting code is up to 2.9X faster

Maintains portability of code:

- Original code runs everywhere
- Transformed code SIMD-friendly

Using Concurrent Links on BG/L



How can system software help?

- Annotations provide a way to maintain portability while adding information needed by the compiler (BG/L isn't the only system that needs these annotations; drops in performance can be seen on other systems)
  - Natural to share transformations (both successes and failures)
  - Other annotations can express more complex properties and/or recommended transformations
- Multiple sends may require special ordering
  - Need graph coloring, similar to data structure decomposition support
  - Should be consistent with physical and logical topology
- System software support can:
  - Identify performance defect (achieving lower than expected performance)
  - Support software aids to performance