Global HPCC Benchmarks in Chapel:
STREAM Triad, Random Access, and FFT

HPC Challenge BOF, SC06
Class 2 Submission
November 14, 2006

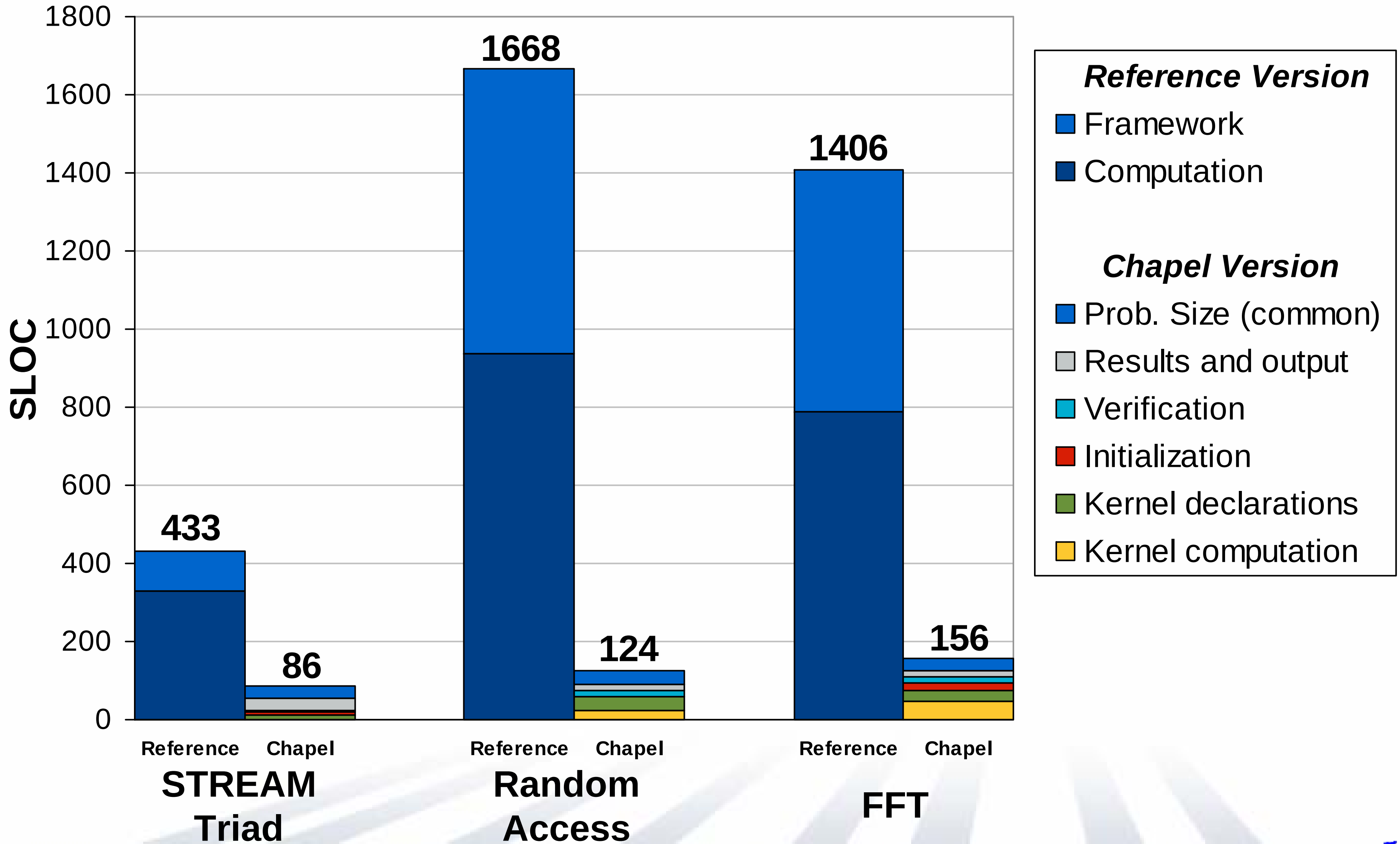
Brad Chamberlain, Steve Deitz,
Mary Beth Hribar, Wayne Wong
Chapel Team, Cray Inc.

Overview

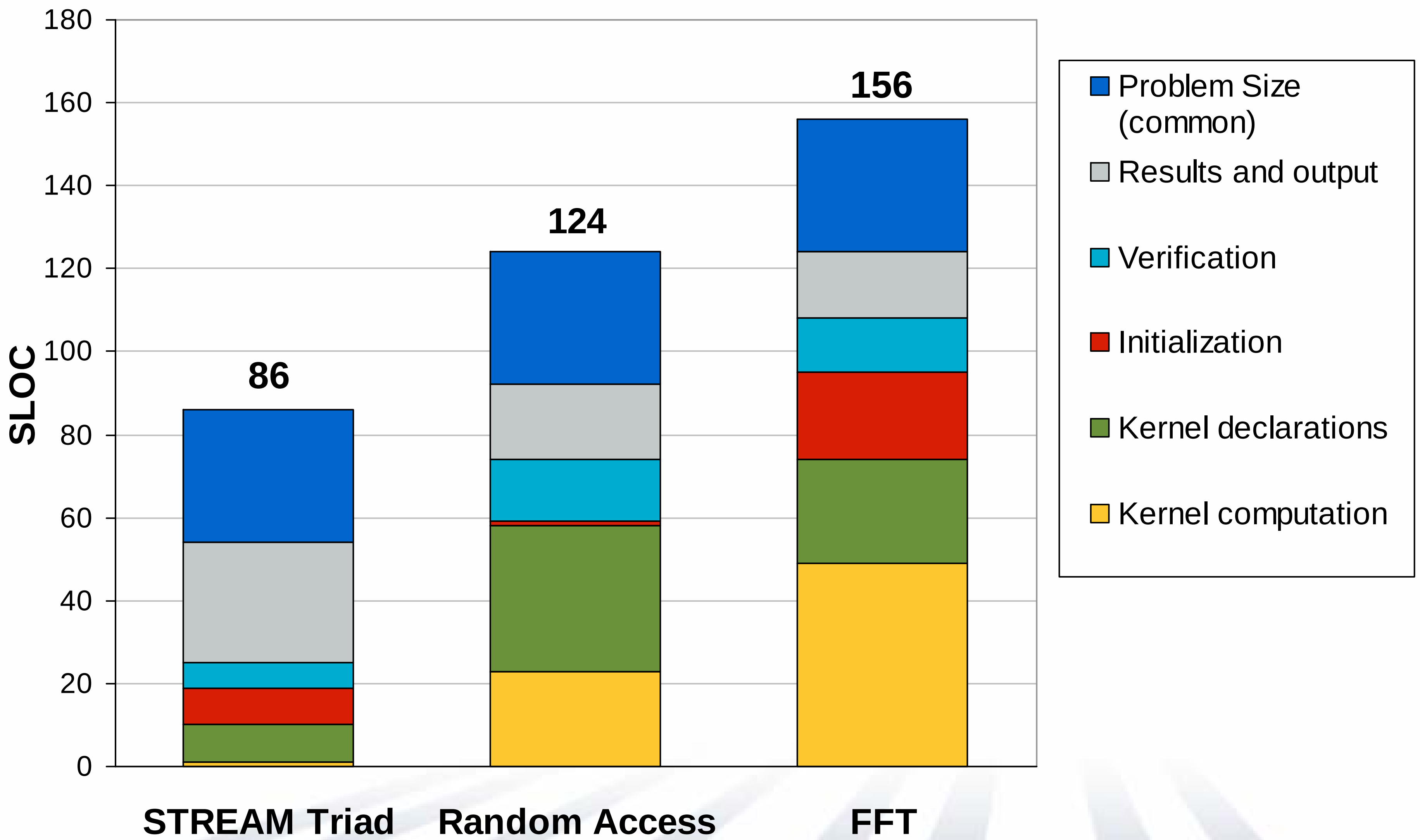
- **Chapel:** Cray's HPCS language

- Our approach to the HPC Challenge codes:
  - performance-minded
  - clear, intuitive, readable
  - general across…
    - types
    - problem parameters
    - modular boundaries

Code Size Summary



Chapel Code Size Summary



STREAM Triad Overview

```
const ProblemSpace: domain(1) distributed(Block) = [1..m];
var A, B, C: [ProblemSpace] elemType;

A = B + alpha * C;
```

STREAM Triad Overview

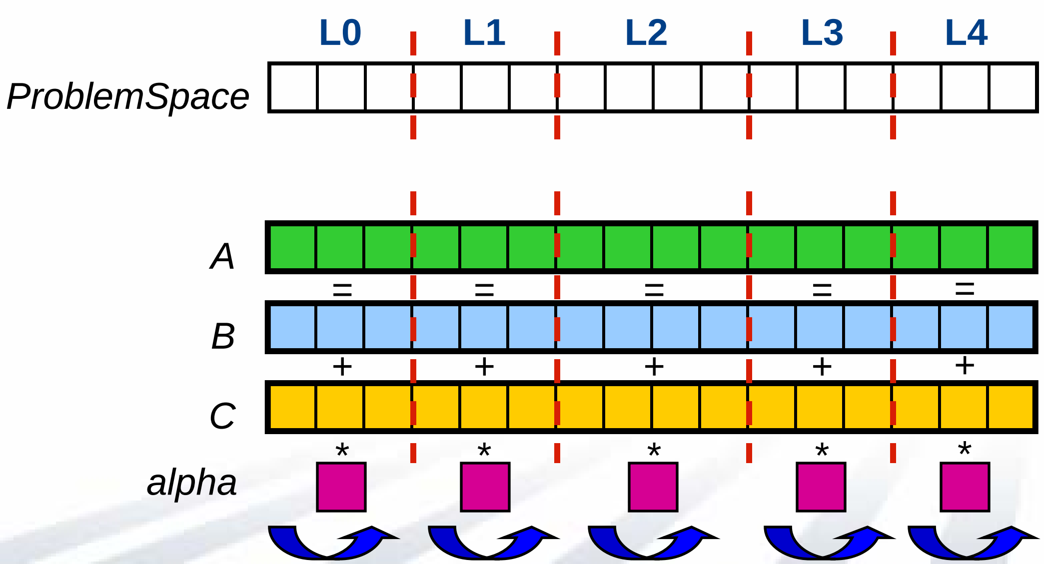
```
const ProblemSpace: domain(1) distributed(Block) = [1..m];
var A, B, C: [ProblemSpace] elemType;

A = B + alpha * C;
```

Declare a 1D arithmetic *domain* (first-class index set)

Specify its distribution

Use domain to declare distributed arrays

Express computation using *promoted* scalar operators and whole-array references ⇒ parallel computation



Random Access Overview

```
[i in TableSpace] T(i) = i;

forall block in subBlocks(updateSpace) do
  for r in RAStream(block.numIndices, block.low) do
    T(r & indexMask) ^= r;
```

Random Access Overview

```
[i in TableSpace] T(i) = i;

forall block in subBlocks(updateSpace) do
  for r in RAStream(block.numIndices, block.low) do
    T(r & indexMask) ^= r;
```

Initialize table using a *forall* expression

Express table updates using *forall*- and for-loops

Random stream expressed modularly using an *iterator*

```
iterator RAStream(numvals,
                  start:randType = 0): randType {
  var val = getNthRandom(start);
  for i in 1..numvals {
    getNextRandom(val);
    yield val;
  }
}
```

FFT Overview (radix 4)

```
for i in [2..log2(numElements)) by 2 {
  const m = span*radix, m2 = 2*m;

  forall (k,k1) in (Adom by m2, 0..) {
    var wk2 = …, wk1 = …, wk3 = …;

    forall j in [k..k+span) do
      butterfly(wk1, wk2, wk3, A[j..j+3*span by span]);

    wk1 = …; wk3 = …; wk2 *= 1.0i;

    forall j in [k+m..k+m+span) do
      butterfly(wk1, wk2, wk3, A[j..j+3*span by span]);
  }
  span *= radix;
}

def butterfly(wk1, wk2, wk3, inout A:[1..radix]) { … }
```

FFT Overview (radix 4)

```
for i in [2..log2(numElements)) by 2 {
  const m = span*radix, m2 = 2*m;

  forall (k,k1) in (Adom by m2, 0..) {
    var wk2 = …, wk1 = …, wk3 = …;

    forall j in [k..k+span) do
      butterfly(wk1, wk2, wk3, A[j..j+3*span by span]);

    wk1 = …; wk3 = …; wk2 *= 1.0i;

    forall j in [k+m..k+m+span) do
      butterfly(wk1, wk2, wk3, A[j..j+3*span by span]);
  }
  span *= radix;
}

def butterfly(wk1, wk2, wk3, inout A:[1..radix]) { … }
```

Parallelism expressed using nested forall-loops

Support for complex and imaginary math simplifies FFT arithmetic

Generic arguments allow routine to be called with complex, real, or imaginary twiddle factors

Chapel Compiler Status

- All codes compile and run with our current Chapel compiler
  - focus to date has been on…
    - prototyping Chapel, not performance
    - targeting a single *locale*
  - platforms: Linux, Cygwin (Windows), Mac OS X, SunOS, …
- No meaningful performance results yet
  - written report contains performance discussions for our codes
- Upcoming milestones
  - **December 2006:** limited release to HPLS team
  - **2007:** work on distributed-memory execution and optimizations
  - **SC07:** intend to have publishable performance results for HPCC`07

Summary

- Have expressed HPCC codes attractively
  - clear, concise, general
  - express parallelism, compile and execute correctly on one locale
  - benefit from Chapel's global-view parallelism
  - utilize generic programming and modern SW Engineering principles

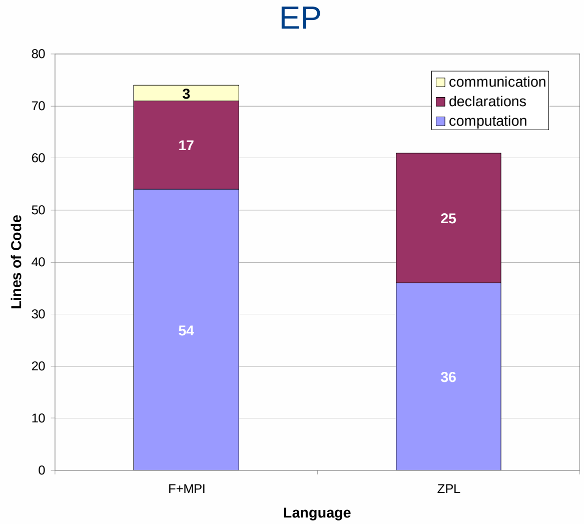
- Our written report contains:
  - complete source listings
  - detailed walkthroughs of our solutions as Chapel tutorial
  - performance notes for our implementations

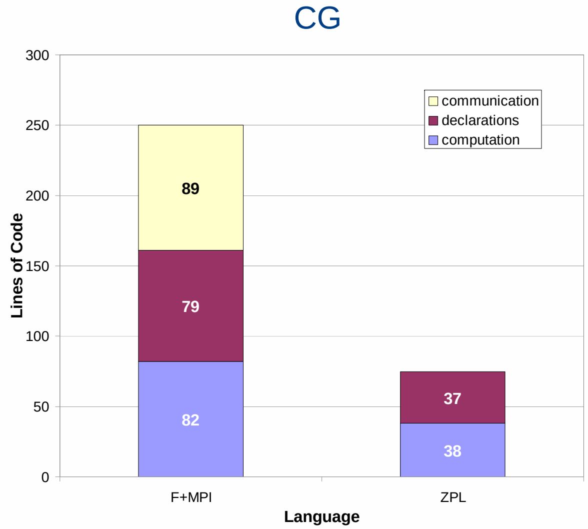
- Report and presentation available at our website:
  http://chapel.cs.washington.edu

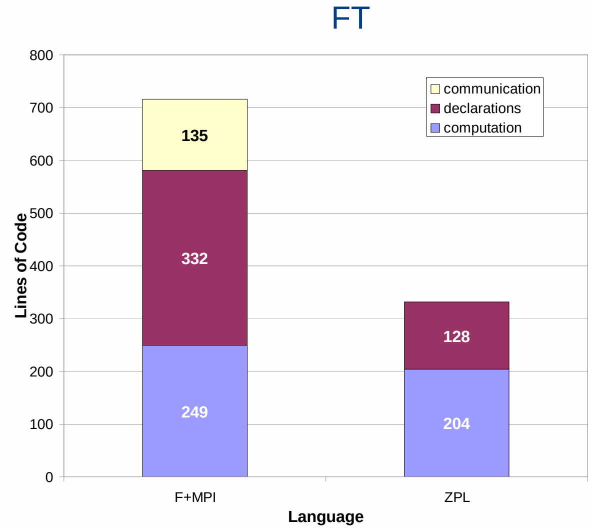
- We're interested in your feedback:
  chapel_info@cray.com

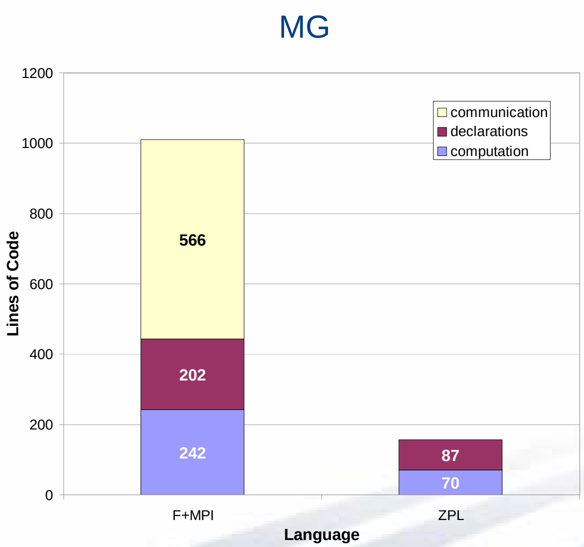
Backup Slides

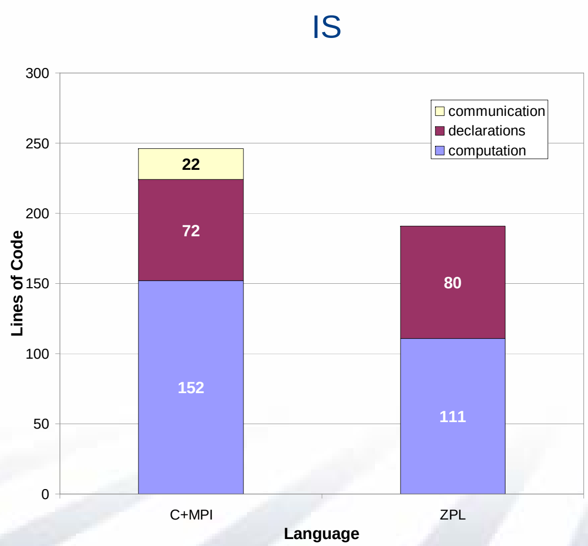
Compact High-Level Code…







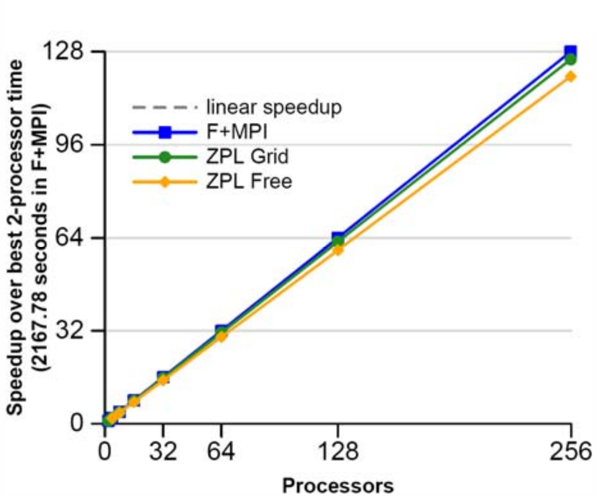




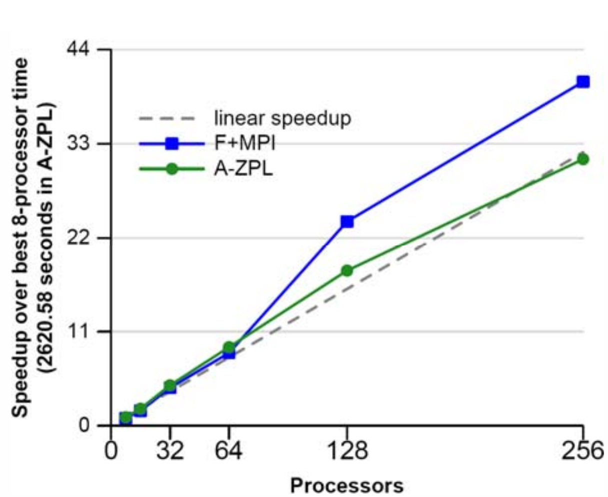
- communication
- declarations
- computation

…need not perform poorly

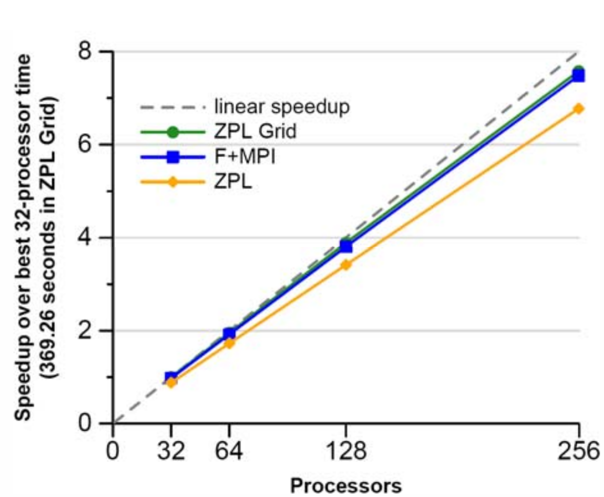
EP



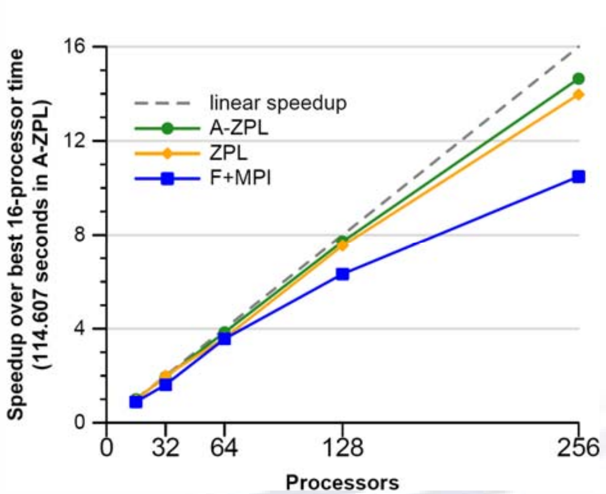
CG



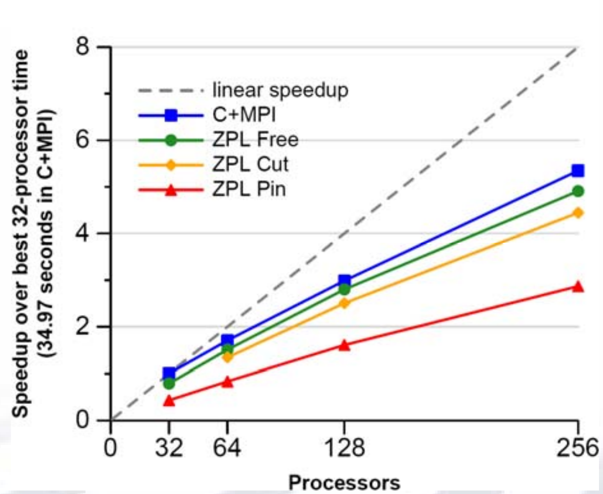
FT

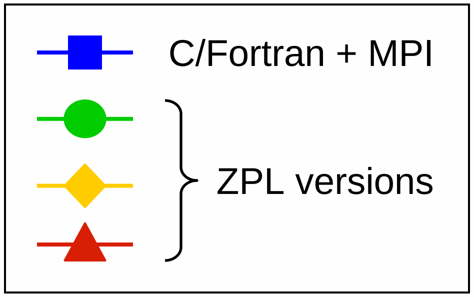


MG



IS





See also Rice University's recent D-HPF work…