Performance modeling of the HPCG benchmark

Vladimir Marjanovic, Jose Gracia and Colin W. Glass

HPC Systems

- Ranking supercomputers

- Two approaches:
  - single application (kernel) :
    HPL, HPCG
  - many applications (kernels):
    NAS benchmark, HPC Challenge, etc

HPL benchmark and TOP500

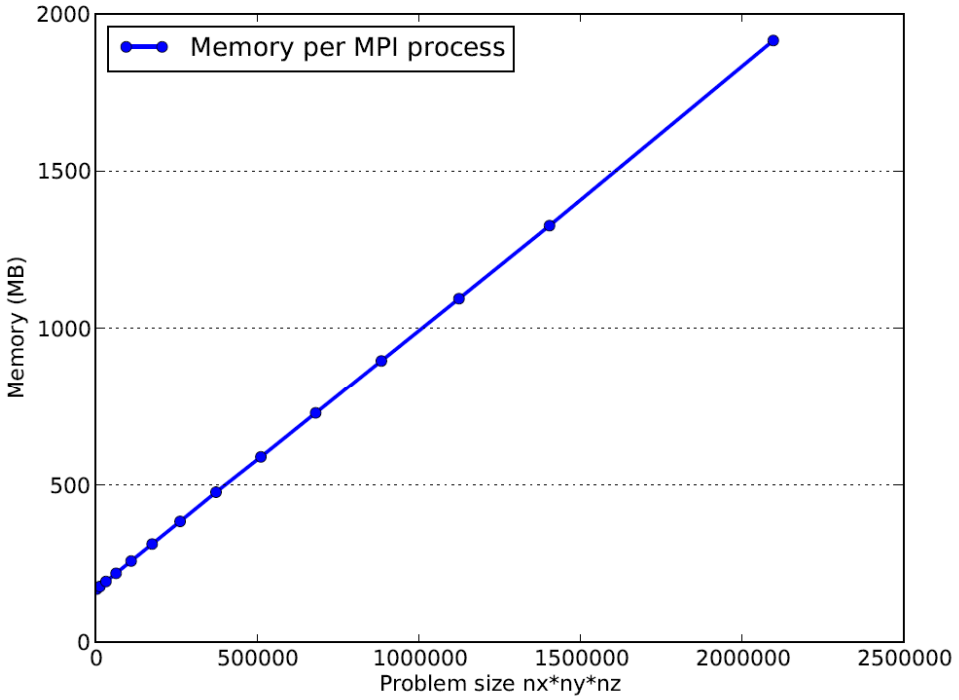
- HPL is de facto the most important benchmark for ranking supercomputers
- Since 1993 TOP 500 uses HPL (first version 1979)
- GFLOP/s is the metric
- REPRESENTATIVITY is an issue!

HPCG History

- First version in September 2013
- Conjugate Gradients solver
- MG preconditioner (from version 2.0 onwards)
- Aims at high representativity for real world applications

HPCG

- MPI and MPI/OpenMP, std lib
- Input: (nx,ny,nz) per MPI process
- Metric: GFLOP/s
- Official run > 3600sec
- Computational complexity $O(n^3)$
  communication complexity $O(n^2)$



MemoryUsage = C1 + C2 * n^3

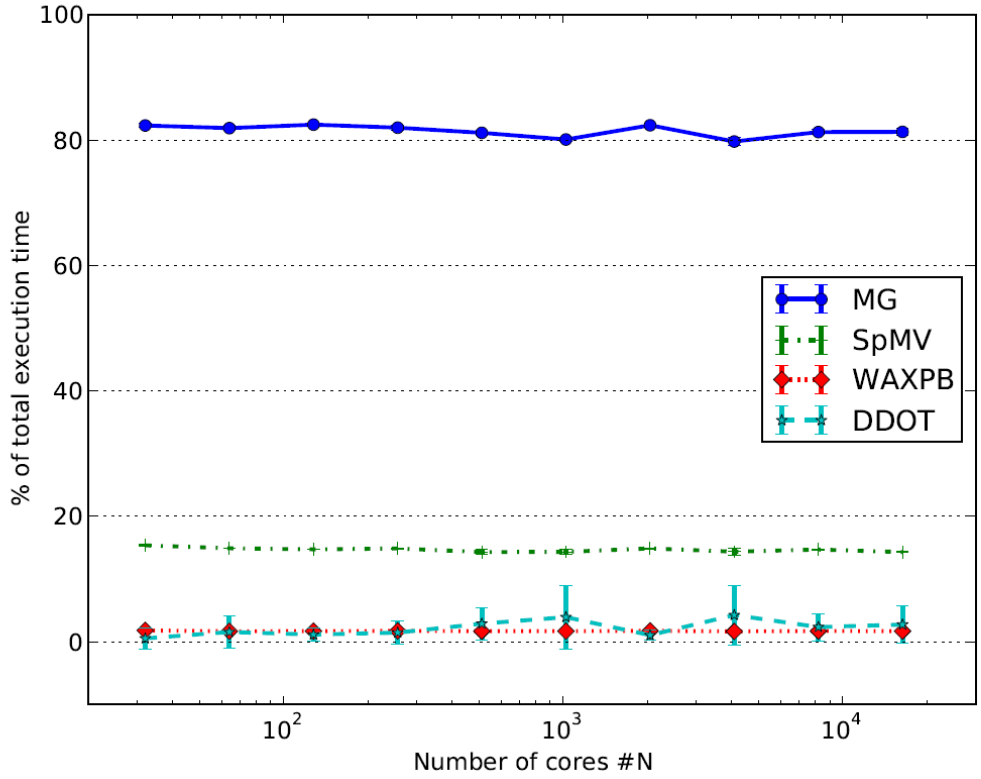
Pseudo-code and % of routines for large problem size

```
for ( i = 0; i<50 && normr>err; i++ ){
  MG(A,r,z);
  DDOT( r ,t ,rtz );
  Allreduce ( rtz );

  if( i > 1 )
    beta = rtz/rtzold;
    WAXPBY( z, beta, p );

  ExchangeHalos( A, p);
  SpMV( A, p, Ap );
  DDOT ( p, Ap, pAp );
  Allreduce ( pAp);
  alpha =rtz/pAp;
  WAXPBY( x, alpha, p);
  WAXPBY( r, -alpha, Ap);
  DDOT( r, r, normr );
  Allreduce (normr);

  normr = sqrt( normr);
}
```



Pseudo-code: Computation and communication routines

```
for ( i = 0; i<50 && normr>err; i++ ){
  MG(A,r,z);
  DDOT( r ,t ,rtz );
  Allreduce ( rtz );

  if( i > 1 )
    beta = rtz/rtzold;
    WAXPBY( z, beta, p );

  ExchangeHalos( A, p);
  SpMV( A, p, Ap );
  DDOT ( p, Ap, pAp );
  Allreduce ( pAp);
  alpha =rtz/pAp;
  WAXPBY( x, alpha, p);
  WAXPBY( r, -alpha, Ap);
  DDOT( r, r, normr );
  Allreduce (normr);

  normr = sqrt( normr);
}
```

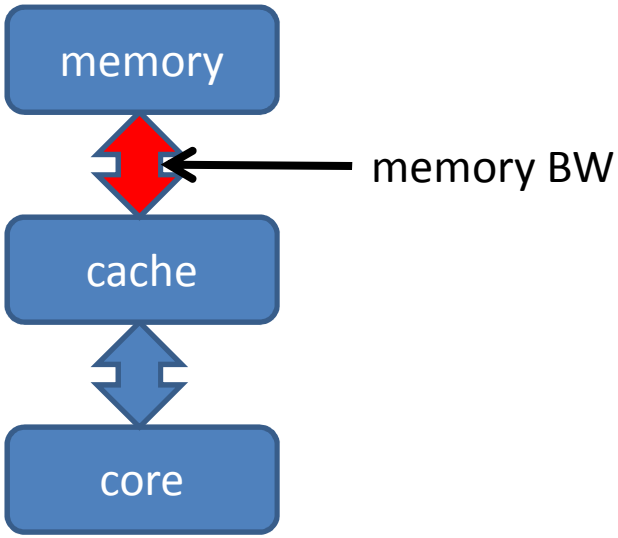
```
/*MG routine*/

if( depth <3){
     ExchangeHalos( );
     SYMGS( );
     ExchangeHalos( );
     SpMV( );
     MG( depth++ )
     ExchangeHalos( );
     SYMGS( );
}else{
     ExchangeHalos( );
     SYMGS( );
}
```

- **Computation routines:**
  - SYMGS
  - SpMV
  - WAXPBY
  - DDOT
- **Communication routines:**
  - Allreduce
  - ExchangesHalos

Computation: Memory/Compute bound – Byte/FLOP

benchmark	kernel	Byte/FLOP
HPL	DGEMM	12/n
HPCG	SpMV, SYMGS	> 4



- Modern hardware ≈ 0.3 Byte/Flop
  e.g E2680v3 has 0.14 Byte/Flop

- HPCG kernels are memory bound on modern hardware

Computational routines

SpMV & SYMGS

```
for (i=0; i<n^3;i++)
  for(j=0; j<27;j++)
    a+=b[i][j]*c[index[i][j]]
```

SpMV and SYMGS have the same computational behavior

WAXPB&DDOT

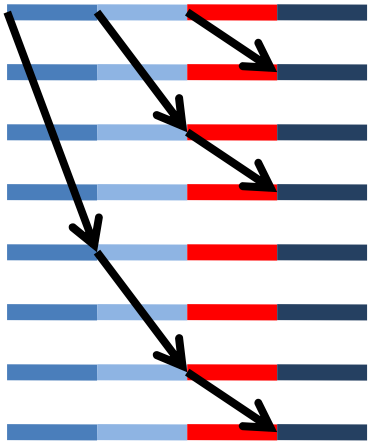
```
for (i=0; i<n^3;i++)
    a[i]=alpha*b[i]+beta*c[j]
```

WAXPB and DDOT 1D loop

$$executionCompRoutine(sec) = \frac{MemoryUsage(Byte)}{BW_{eff}(Byte/sec)}$$

Communication: MPI_Allreduce

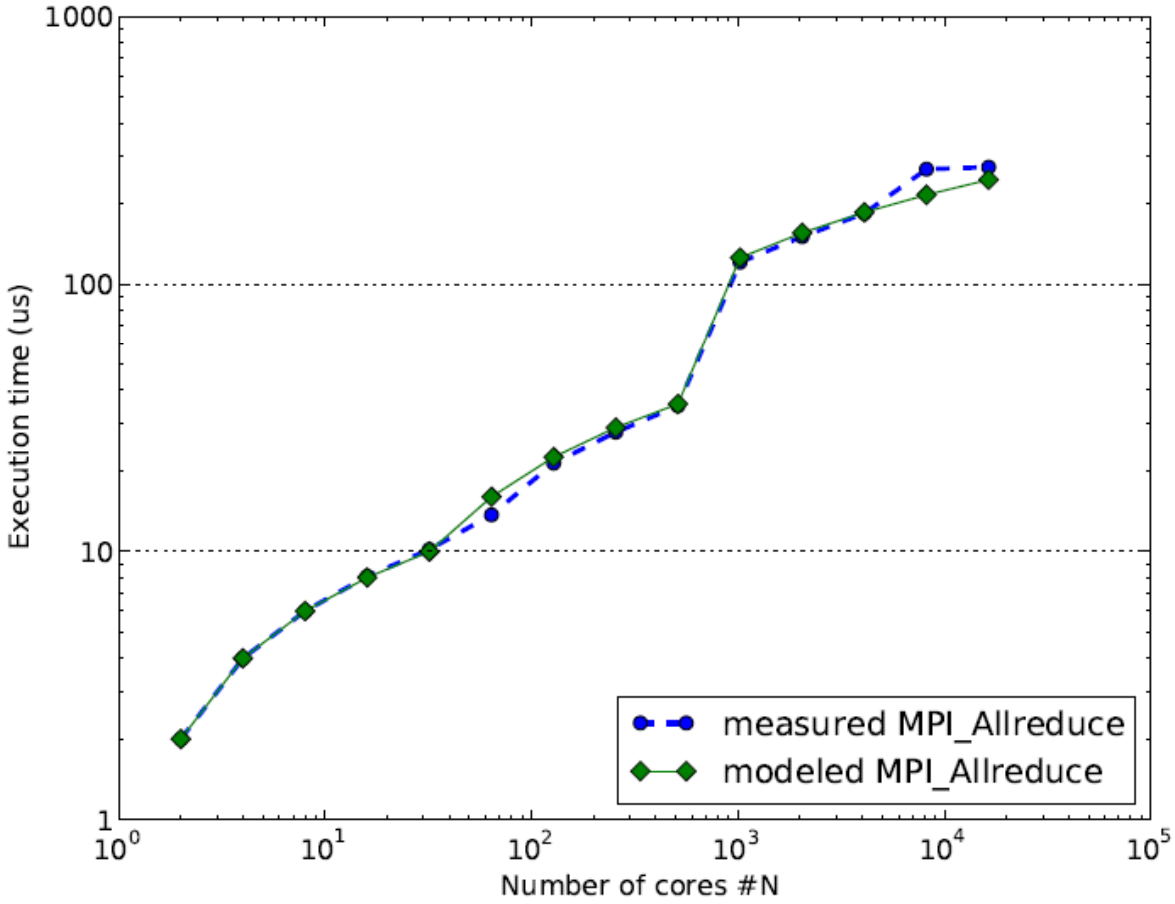
- Hypercube algorithm: O(log(N))
- HPCG calls MPI_Allreduce three times per iteration
- Message size = 8Byte
- k different latency levels: within socket, within node, within blade, within cabinet, etc



hypercube algorithm

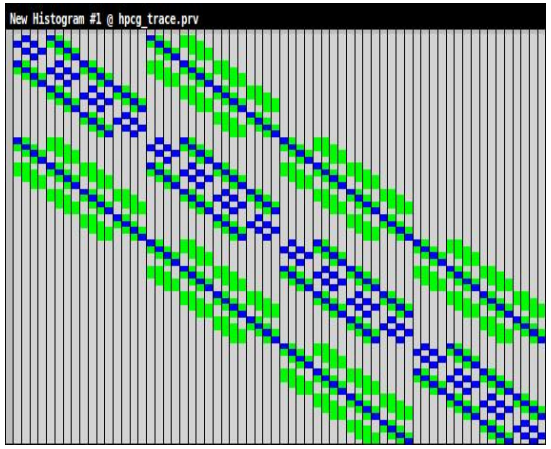
$$executionAllreduce(sec) = \sum_{i=1}^{k} latency_i (\log(M_i) - \log(M_{i-1}))$$

Communication: MPI_Allreduce



Communication: ExchangeHalos

- Exchange halos with neighbors
- Maximum number of neighbors is 26
- For large problem size one process exchange up to 1MB



communication pattern

$$HaloSize(Byte) = (2 \cdot (nx \cdot ny + nx \cdot nz + nz \cdot ny) + 4 \cdot (nx + ny + nz) + 8)(Byte)$$

$$executionHaloEx(sec) = \frac{HaloSize(Byte)}{IC\_BW_{eff}\,(Byte/sec)} + overhead(MPIcalls)$$

Whole application

```
for ( i = 0; i<50 && normr>err; i++ ){
  MG(A,r,z);
  DDOT( r ,t ,rtz );
  Allreduce ( rtz );

  if( i > 1 )
    beta = rtz/rtzold;
    WAXPBY( z, beta, p );

  ExchangeHalos( A, p);
  SpMV( A, p, Ap );
  DDOT ( p, Ap, pAp );
  Allreduce ( pAp);
  alpha =rtz/pAp;
  WAXPBY( x, alpha, p);
  WAXPBY( r, -alpha, Ap);
  DDOT( r, r, normr );
  Allreduce (normr);

  normr = sqrt( normr);
}
```

```
/*MG routine*/

if( depth <3){
    ExchangeHalos( );
    SYMGS( );
    ExchangeHalos( );
    SpMV( );
    MG( depth++ )
    ExchangeHalos( );
    SYMGS( );
}else{
    ExchangeHalos( );
    SYMGS( );
}
```

- Combine routines and sum over execution times
- Execution time is modeled and FLOP are computed, giving FLOP/s

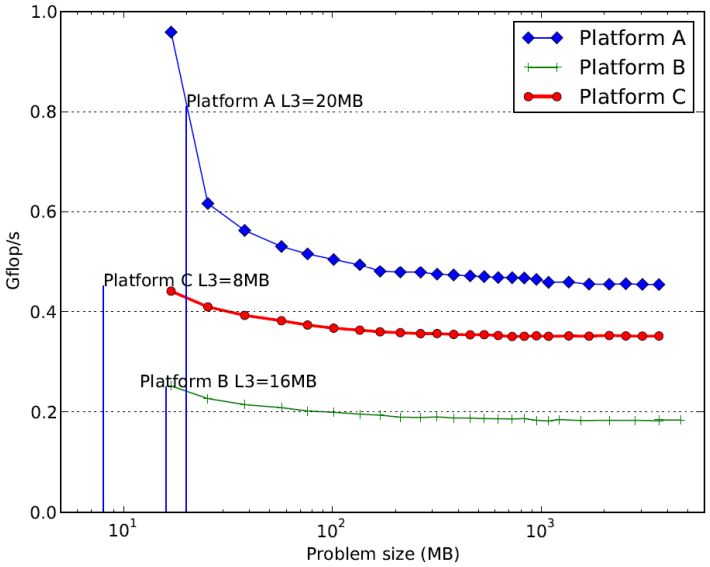
$$executionMG = HaloEx(depth=3) + SYMGS(depth=3) + \sum_{depth=0}^{2}(2\cdot SYMGS(depth) + SpMV(depth) + 3\cdot HaloEx(depth))$$

$$executionHPCG = MG + SpMV(depth=0) + HaloEx(depth=0) + 3\cdot(DDOT + Allreduce + WAXPB)$$

Platforms Software Characterization

	Platform A	Platform B	Platform C
Stream(MB/s)	4705	1700	3430
Pingpong(µs)	2-4	2-90	4-240

- Small problem size: HPCG avoids memory bandwidth bottleneck
- Large problem size: HPCG performance is proportional to STREAM benchmark

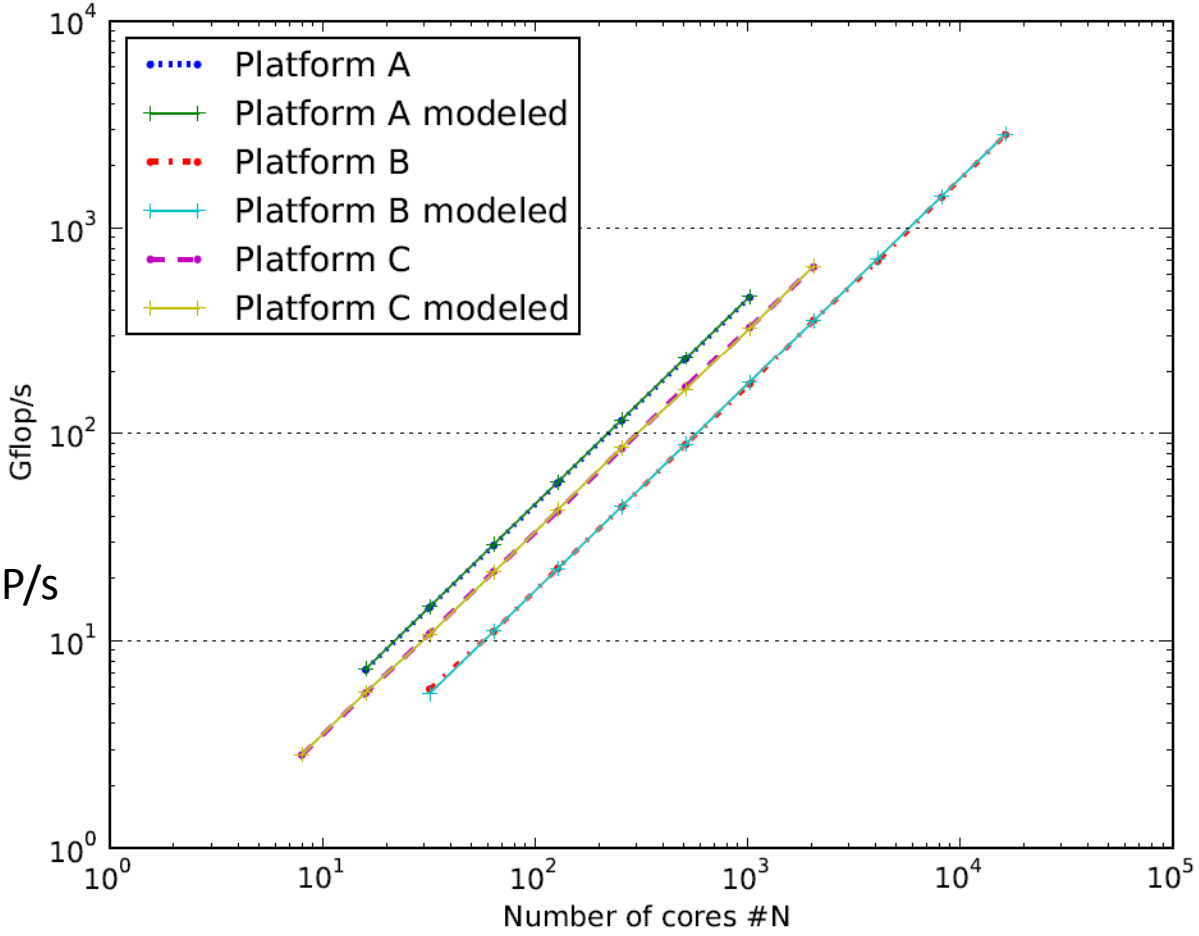


HPCG: GFLOP/s vs. problem size single node

Accuracy of the model

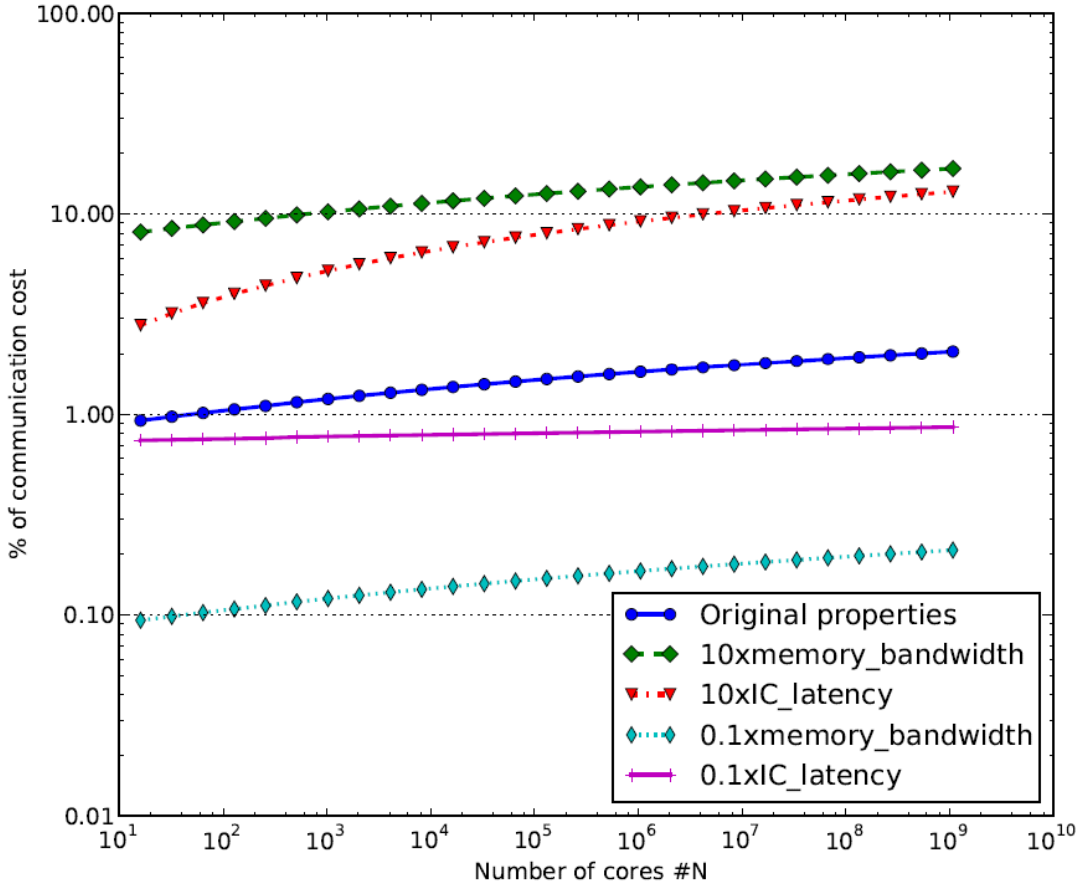
- Accuracy ±2%

- 93600 cores machine
  Official run: 39114GFLOP/s
  Model : 39319GFLOP/s



Performance prediction

- For current hardware communication cost is 3%
- Extrapolation to 1billion core machines



Conclusions

- HPCG model shows high accuracy 2%

- Arbitrary problem size → single property dominates

- Information content of the full system benchmark equals STREAM benchmark on a single node