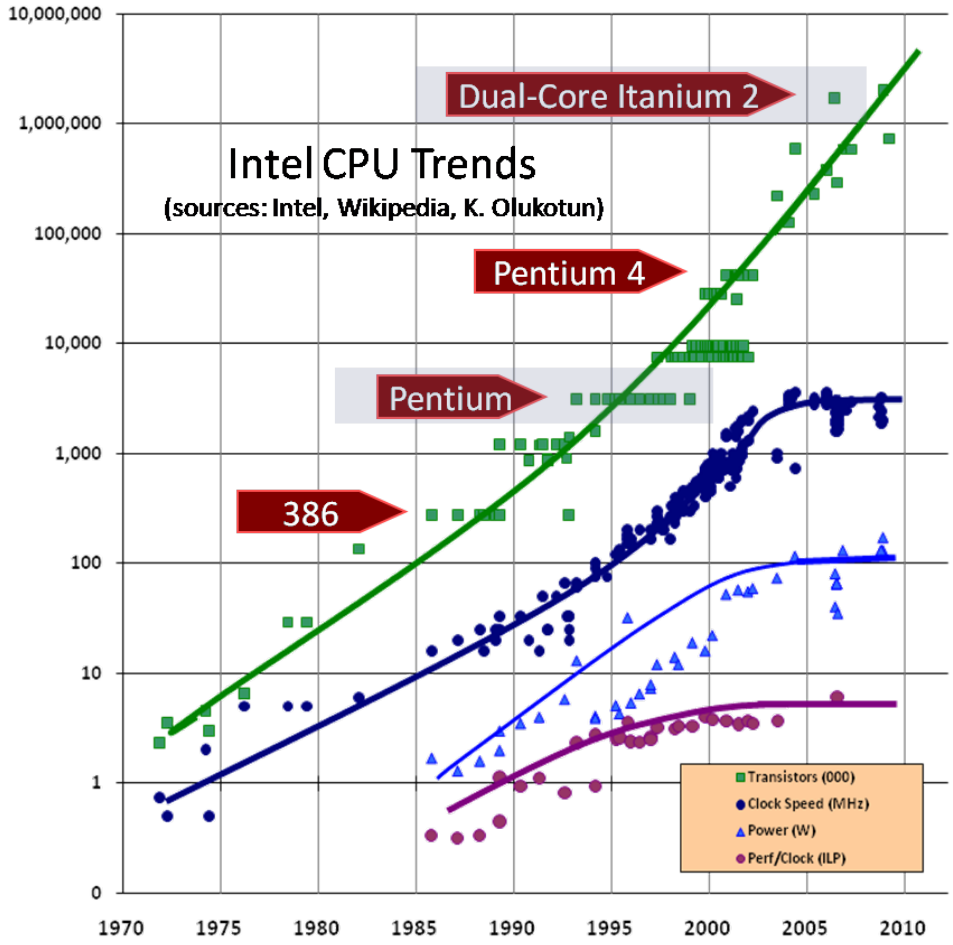
Hybrid Parallelism on BG/Q with OpenMP

February 2, 2015 | T. Hater | EIC — JSC — FZJ

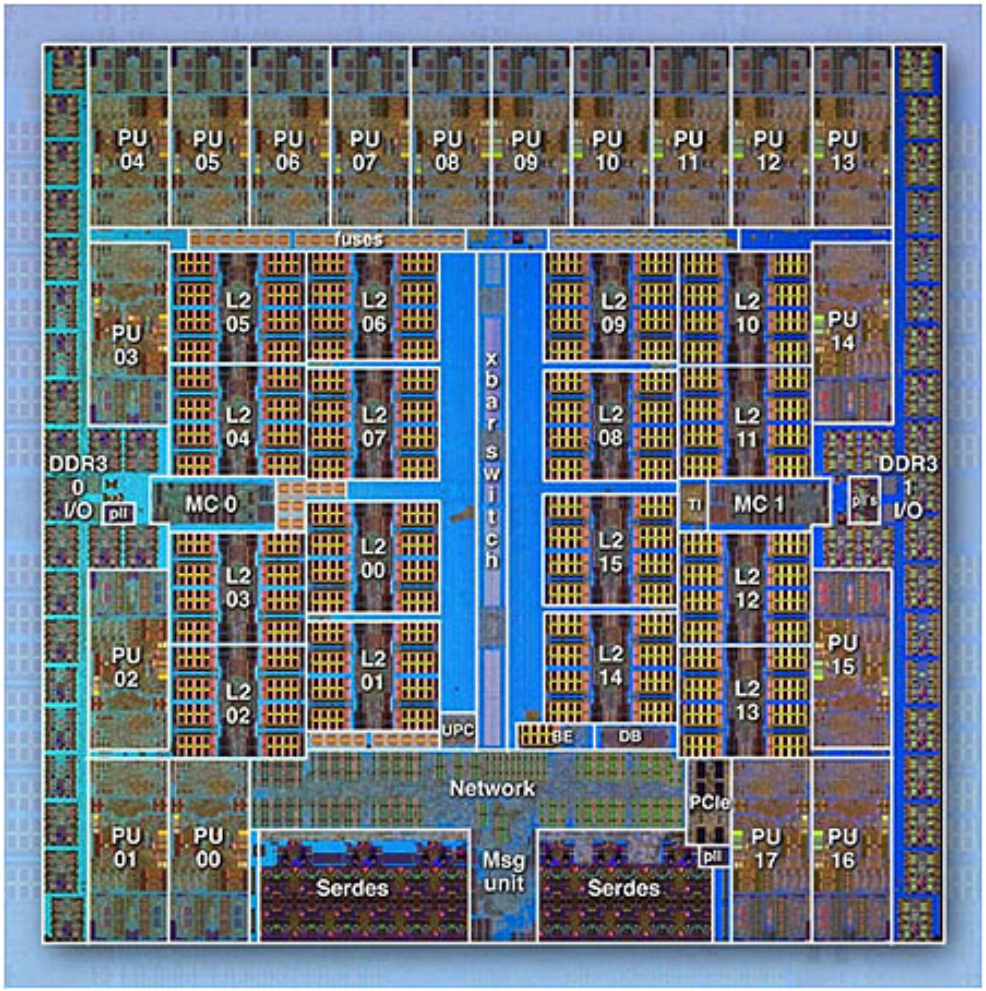
Roadmap

- Motivation: Multi-core CPUs and the BG/Q chip.
- Shared memory and OpenMP overview.
- An example on JUQUEEN.
- Tuning for BG/Q.
- Reference material.

The free lunch is over
Herb Sutter 2009

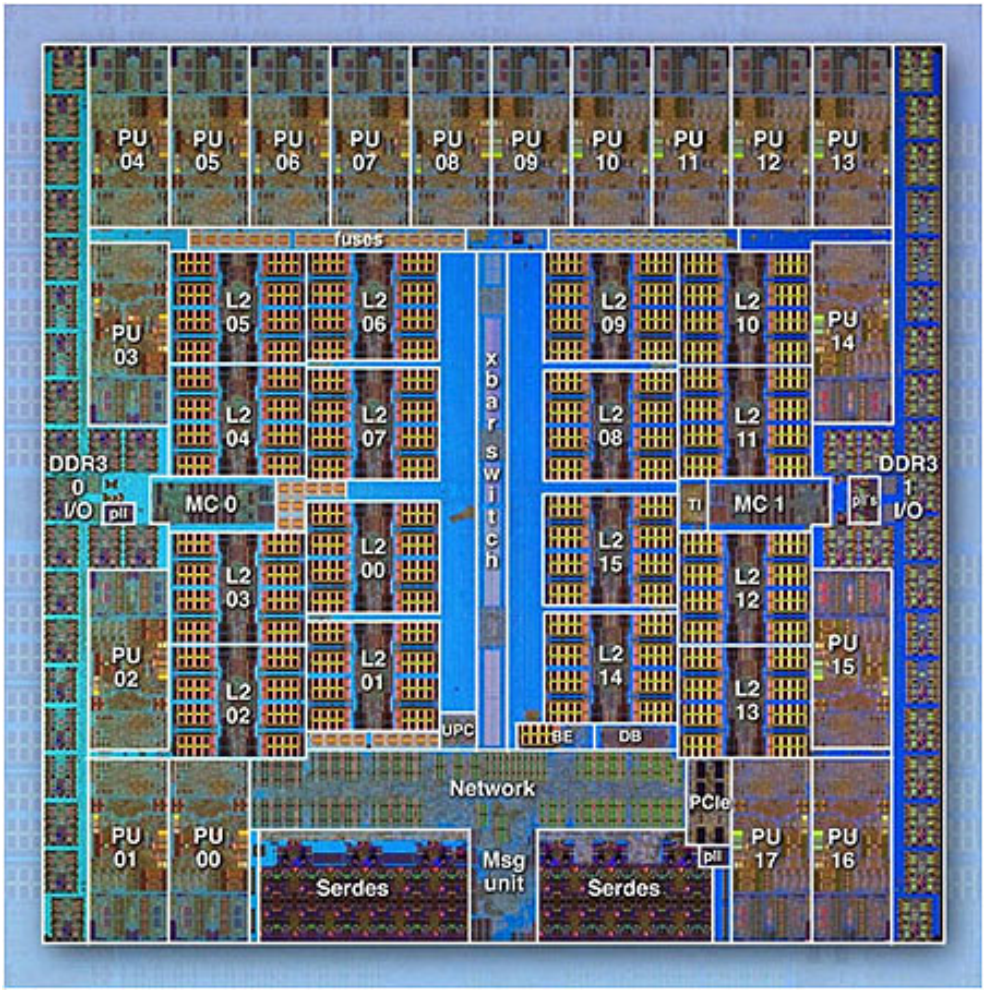


The BG/Q Compute Chip



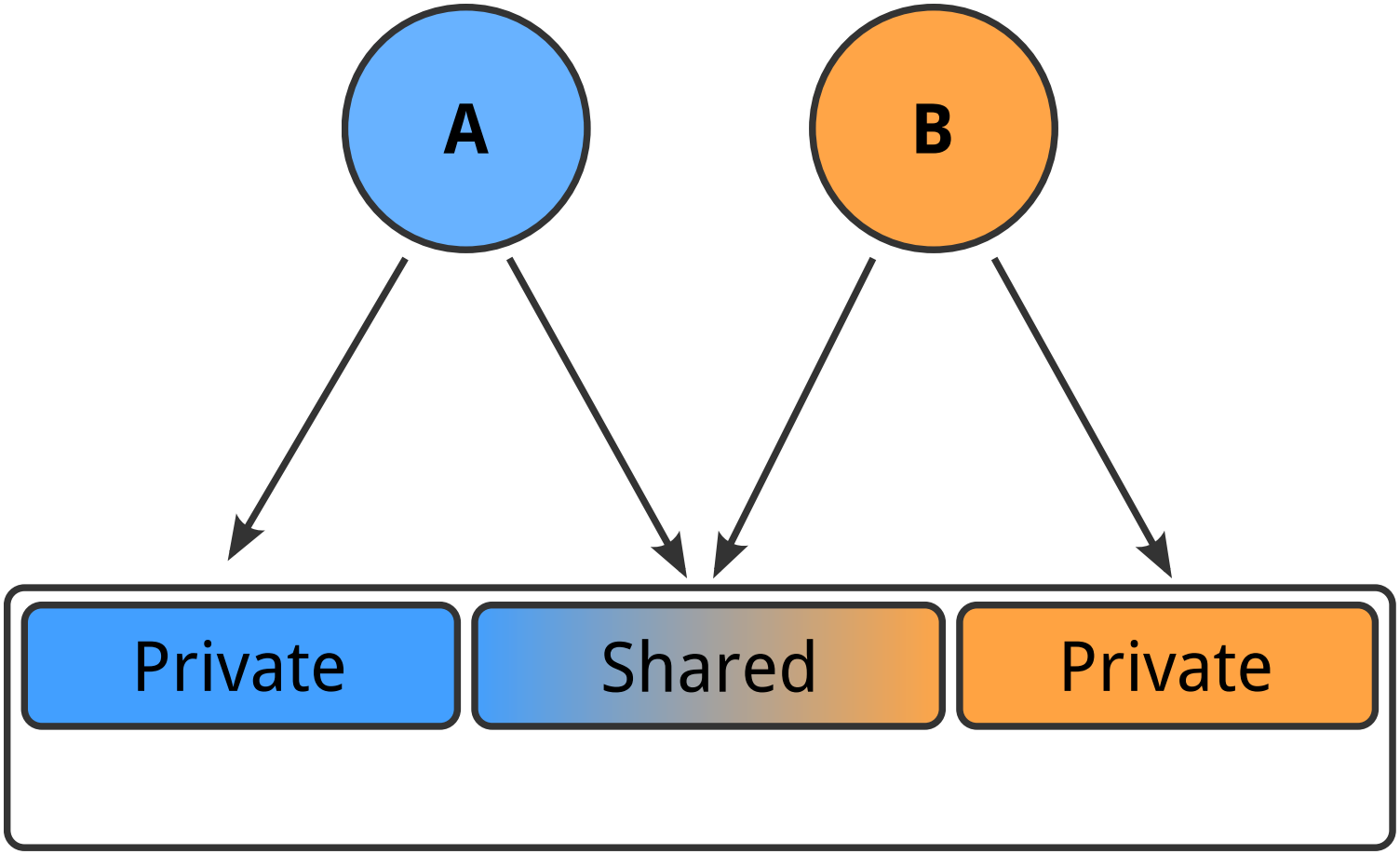
- SoC incl Network
- 1.6 *GHz* in-order
- 200 *GFlop/s*
- 16 GB @ 30 GB/s_{eff}
- 16 kB *L*1\$
  32 MB *L*2\$
- 16 Cores $\times$
  2 Pipelines
- 4-way SMT
- 4-way SIMD

The BG/Q Compute Chip



- SoC incl Network
- 1.6 *GHz* in-order
- 200 *GFlop/s*
- 16 GB @ 30 GB/s_{eff}
- 16 kB *L*1\$
  32 MB *L*2\$
- 16 Cores $\times$
  2 Pipelines
- 4-way SMT
- 4-way SIMD

Shared Memory Parallelism



OpenMP
At a glance

- `pragma` annotations designate regions for parallelism.
- Compiler translates regions into calls to a runtime.
- Runtime manages threads, synchronization.
- Supported by almost all major compilers.
- Works with C, C++ and Fortran
- Just as dangerous as any other option for SMP.

OpenMP
Annotations

- Serial programs may be parallelised *'iteratively'*.
- Can revert to serial by re-compiling.
- Most important cases are simple and efficient
  - `#pragma omp parallel for`
  - `#pragma omp parallel for reduction(...)`
- Support for data management.
- Worksharing can be tuned by further clauses.

An example

- Demonstration of two important techniques.
- Compute π in a two step process.

$$X_n = \frac{(-1)^n}{2n+1} \qquad n = 0 \ldots N-1 \tag{1}$$

$$\frac{\pi}{4} = \text{atan}(1) \overset{N\to\infty}{=} \sum_{n=0}^{N-1} X_n \tag{2}$$

- All examples are compiled with -O2
- One node BG/Q and a single process.

OpenMP
Serial

```
double res = 0.0;
double arr[N];

for(int n = 0; n < N; ++n) {
  arr[n] = ((n&1) ? -1.0 : +1.0)/(2*n + 1);
}

for(int n = 0; n < N; ++n) {
  res += arr[n];
}
```

OpenMP
Slightly less serial

```
double res = 0.0;
double arr[N];
#pragma omp parallel for shared(arr)
for(int n = 0; n < N; ++n) {
  arr[n] = ((n&1) ? -1.0 : +1.0)/(2*n + 1);
}

for(int n = 0; n < N; ++n) {
  res += arr[n];
}
```

OpenMP
Parallel at last

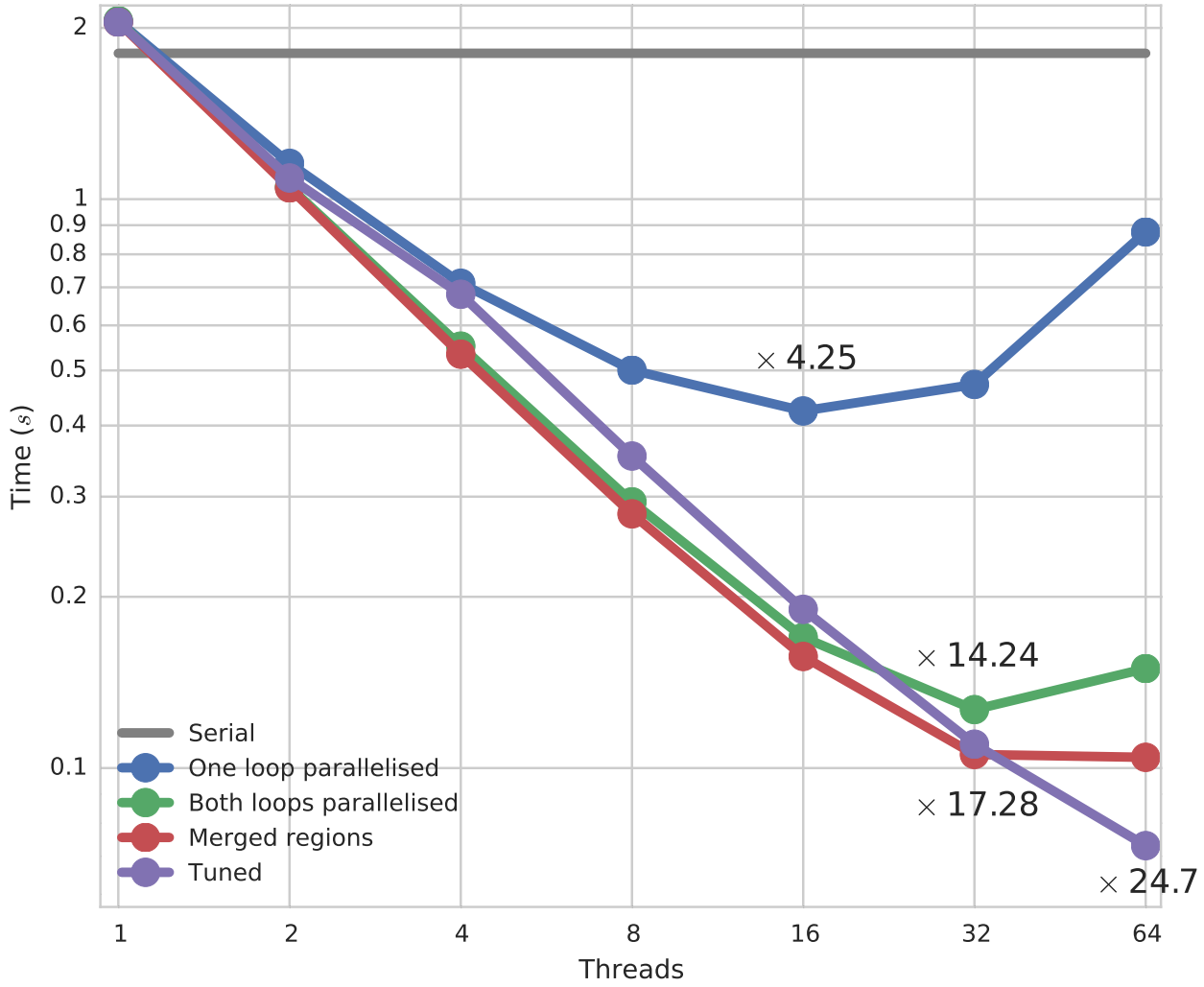
```
double res = 0.0;
double arr[N];
#pragma omp parallel for shared(arr)
for(int n = 0; n < N; ++n) {
  arr[n] = ((n&1) ? -1.0 : +1.0)/(2*n + 1);
}
#pragma omp parallel for shared(arr) reduction(+:res)
for(int n = 0; n < N; ++n) {
  res += arr[n];
}
```

OpenMP
Optimized

```
double res = 0.0;
double arr[N];
#pragma omp parallel shared(arr) reduction(+:res)
{
  #pragma omp for
  for(int n = 0; n < N; ++n) {
    arr[n] = ((n&1) ? -1.0 : +1.0)/(2*n + 1);
  }
  #pragma omp for
  for(int n = 0; n < N; ++n) {
    res += arr[n];
  }
}
```

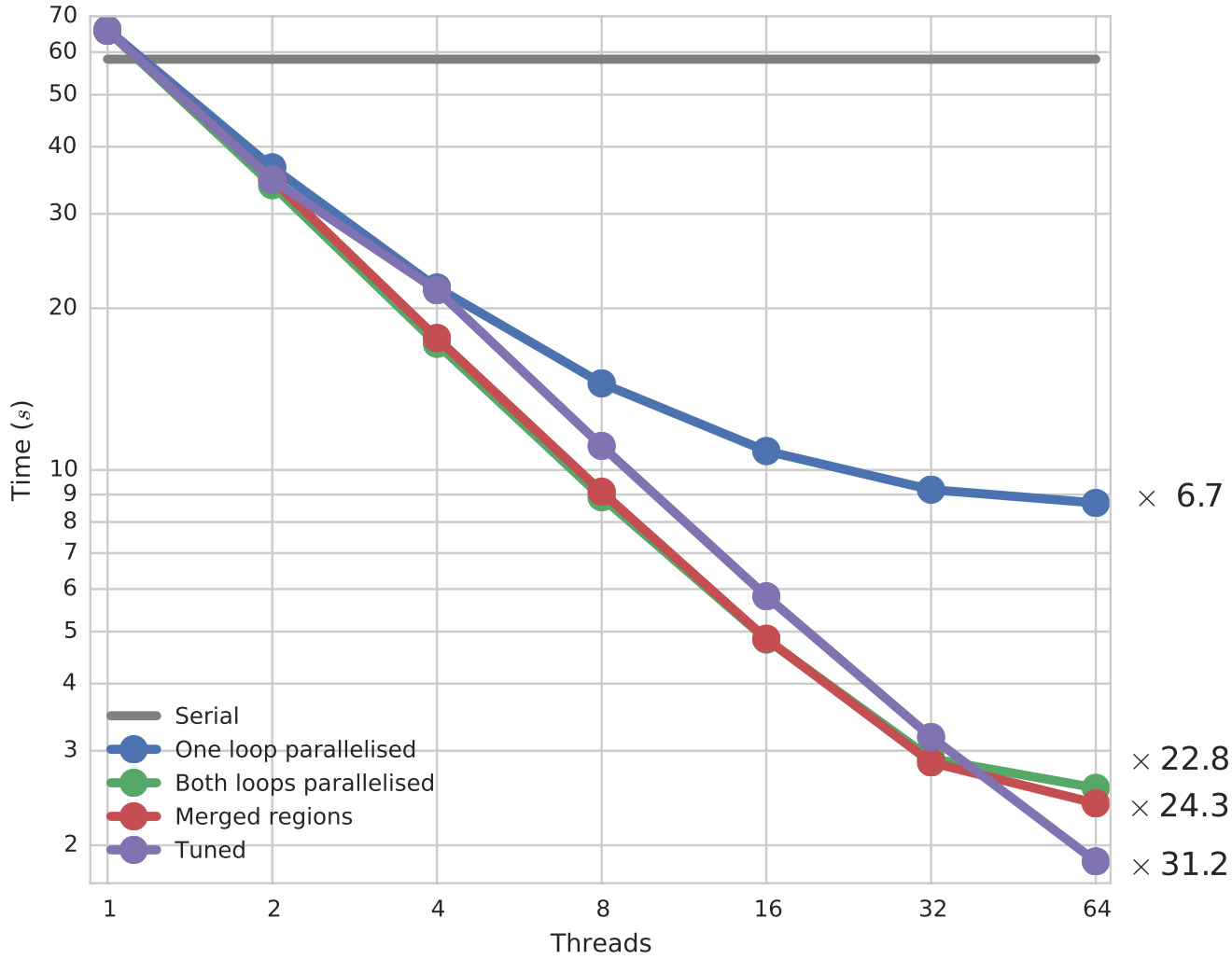
Results

$N = 2^{15}$, 1000 Repetitions



Results

$N = 2^{20}$, 1000 Repetitions



OpenMP Support on BG/Q

- `mpixlc_r -qsmp=omp`
- Only standard OpenMP pragmas, no auto parallelisation.
- Enables a set of optimisations (`-qsmp=omp:noopt`).
- OpenMP 3.x only
- Must have `OMP_PROC_BIND=TRUE`
- 16 kB L1$: Keep your loops small.

OpenMP Support on BG/Q

- `mpixlc_r -qsmp=omp`
- Only standard OpenMP pragmas, no auto parallelisation.
- Enables a set of optimisations (`-qsmp=omp:noopt`).
- OpenMP 3.x only
- Must have `OMP_PROC_BIND=TRUE`
- 16 kB L1$: Keep your loops small.

Tuning for BG/Q
Environment variables

- Standard options
  - `OMP_NUM_THREADS=<n>`
  - `OMP_WAIT_POLICY=ACTIVE`
- XLC specific
  - `XLSMPOPTS=...:...:...`
  - `parthds=<n>` Number of threads
  - `spins=<s>:yields=<y>` Waiting policy
  - `start=<p0>:stride=<pp>` Core binding
  - `BG_SMP_FAST_WAKEUP=YES` Use HW support
  - `BG_THREADLAYOUT=1|2` High-level core binding

Tuning for BG/Q
Comm threads

- May make progress on MPI asynchronously and increase message throughput.
- Initialize MPI with `MPI_THREAD_MULTIPLE`
  or
  Export `PAMID_ASYNC_PROGRESS=1`
- The rest should happen *automagically*

BG/Q Beyond OpenMP

- Transactional Memory (TM)
- Speculative Execution (SE)
- L2 Atomics, Locks and Barriers
- IBM ESSL library with build-in SMP
  - `-lesslsmpbg`
  - Contains most of BLAS/LAPACK/FFTW

Resources

- OpenMP Specification v. 3.x

  `openmp.org`

- IBM XL{C, CXX, F} for BG/Q compiler docs

  `http://www-01.ibm.com/support/knowledgecenter/SS2LWA/welcome`

- IBM redbooks

  `http://www.redbooks.ibm.com/redbooks/pdfs/sg247948.pdf`

- Argonne BG/Q wiki

  `https://www.alcf.anl.gov/user-guides/tuning-mpi-bgq`

  `https://wiki.alcf.anl.gov/parts/index.php/Blue_Gene/Q`

Questions?

I will be around in the hands-on sessions, too