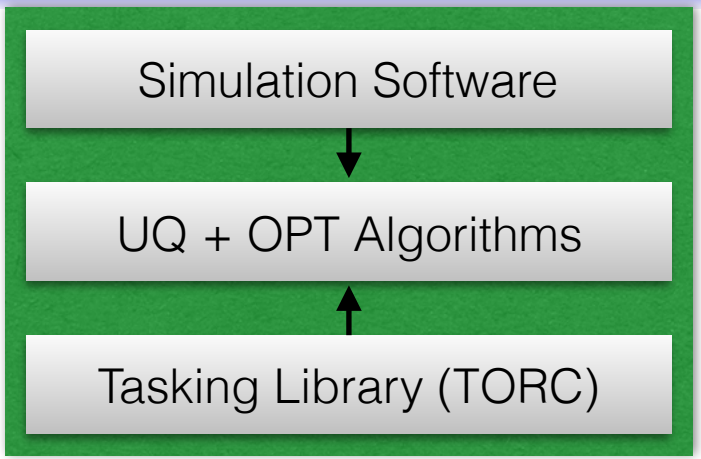
HPCSE - II

«MPI Programming Model: MPI Tasking Library »

Panos Hadjidoukas

Π4U + TORC

- Applications + Algorithms + HPC
- TORC: tasking library



- All the algorithms in Π4U (e,g, CMAES, TMCMC, ABC-SubSim) are TORC-based applications
  - tasks are spawned and submitted for execution to a set of workers
  - common development approach: serial → OpenMP → TORC

TORC Tasking Library

- Execution model: an application consists of multiple MPI processes with private memory, running on cluster nodes
  - Each MPI process consists of 1 or more worker threads
- Task: (remote) execution of a function on a set of data that are passed as arguments
  - Tasks are executed asynchronously and in any order
  - No data dependencies or point-to-point communication
  - Parent-child relationship, arbitrarily nested

Sequential Code

```
void task(double *x, double *y) {
      *y = x[0]+x[1];
}

int main(int argc, char *argv[]) {
      double result[100];

      for (int i=0; i<100; i++) {
          double d[2];
          d[0] = drand48();
          d[1] = drand48();
          task(d, &result[i]);
      }

    /* print results */
    return 0;
}
```

OpenMP Code

```
void task(double *x, double *y) { *y = x[0] + x[1]; }

int main(int argc, char *argv[]) {
      double result[100];

      #pragma omp parallel
      #pragma omp single nowait
      {
          for (int i=0; i<100; i++) {
              double d[2];
              d[0] = drand48();
              d[1] = drand48();
              #pragma omp task firstprivate(d, i) shared(result)
              {
                task(d, &result[i]);
              }
          }
          #pragma omp taskwait
          /* print results */
      }
   return 0;
}
```

TORC-based Code

```
void task(double *x, double *y) { *y = x[0] + x[1]; }

int main(int argc, char *argv[]) {
      double result[100];
      torc_register_task(task);
      torc_init(argc, argv, MODE_MW);

      for (int i=0; i<100; i++) {
          double d[2];
          d[0] = drand48();
          d[1] = drand48();

          torc_task(-1, task, 2,
              2, MPI_DOUBLE, CALL_BY_COP,    /* IN (PRIVATE COPY) */
              1, MPI_DOUBLE, CALL_BY_RES,    /* OUT */
              d, &result[i]);
      }
      torc_waitall();

      /* print results */
      return 0;
}
```

main(): the primary task of the application

A task function can:

- spawn new TORC tasks
- include OpenMP/MPI code
- call an external application

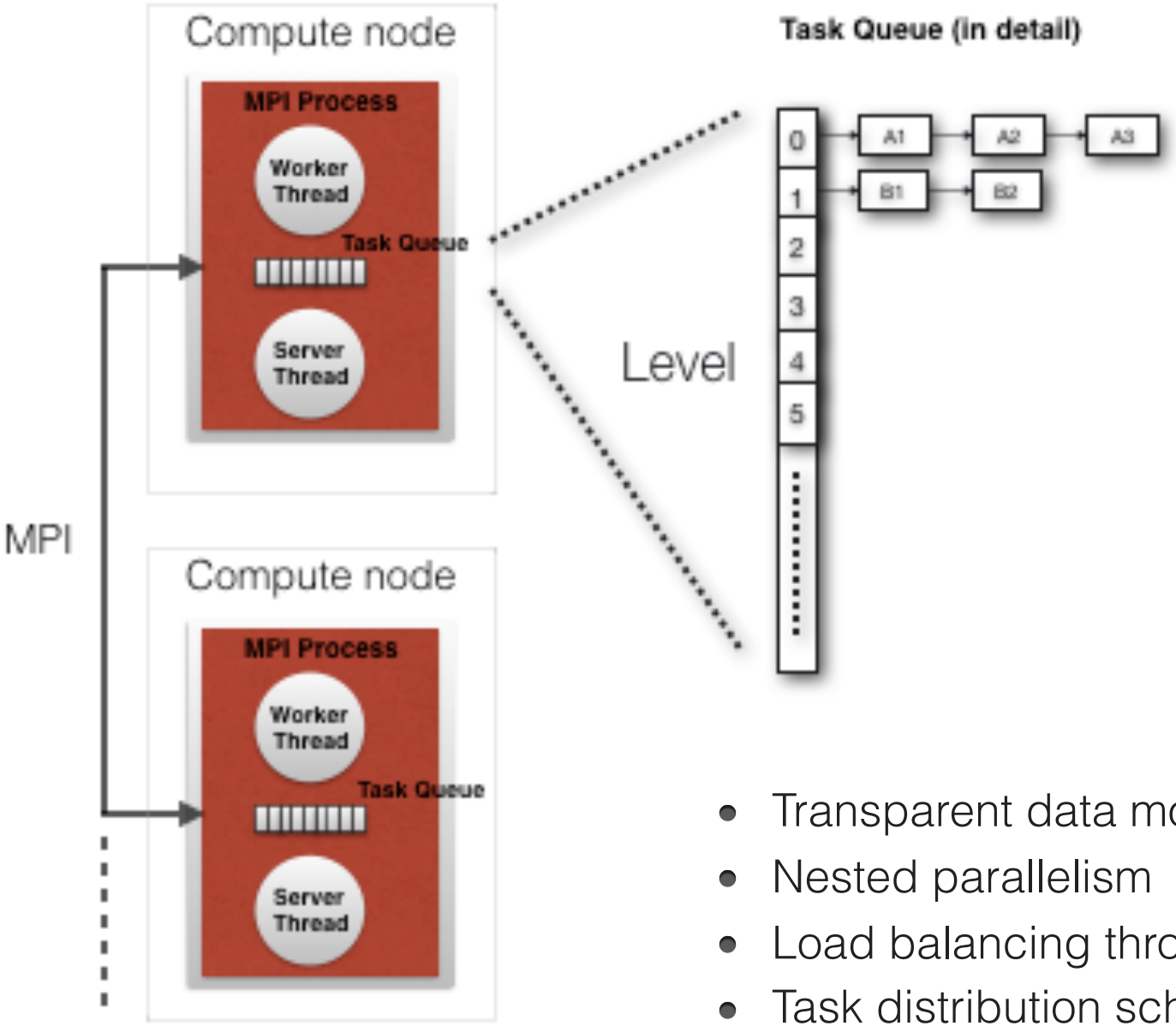
CMA-ES - Sequential C code

```
/* Iterate until stop criterion holds */
while(!cmaes_TestForTermination(&evo))
{
  /* generate lambda new search points, sample population */
  pop = cmaes_SamplePopulation(&evo);

  lambda = cmaes_Get(&evo, "lambda");
  dim = cmaes_Get(&evo, "dim");

  /* evaluate the new search points using fitfun from above */
  for (i = 0; i < lambda; ++i) {
    fitfun(pop[i], dim, &arFunvals[i]);
  }

  /* update the search distribution */
  cmaes_UpdateDistribution(&evo, arFunvals);
  /* read instructions for printing output or
     changing termination conditions */
  cmaes_ReadSignals(&evo, "signals.par");
  step++;
}
```

CMA-ES - TORC tasks

```
torc_init(argn, args, MODE_MS);
...
/* Iterate until stop criterion holds */
while(!cmaes_TestForTermination(&evo))
{
  ...
  /* evaluate the new search points using fitfun from above */
  for (i = 0; i < lambda; ++i) {
    torc_task(-1, fitfun, 3,
      dim, MPI_DOUBLE, CALL_BY_VAL,
      1, MPI_INT, CALL_BY_COP,
      1, MPI_DOUBLE, CALL_BY_RES,
      pop[i], &dim, &arFunvals[i]);
  }
  torc_waitall();
  ...
  step++;
}
```

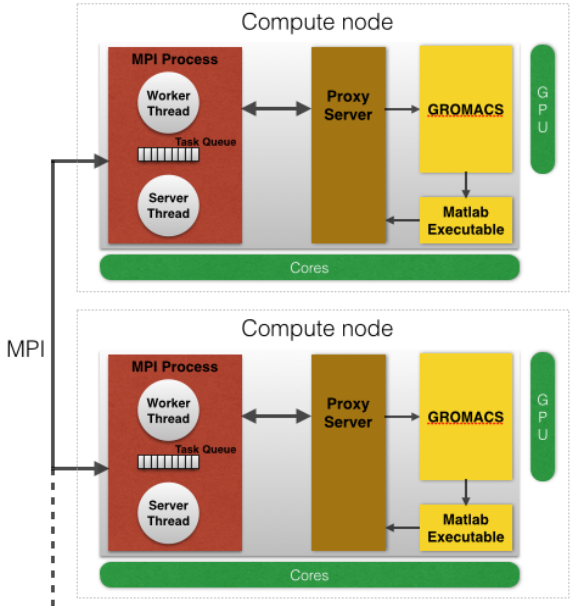
Runtime Architecture



- Transparent data movement
- Nested parallelism
- Load balancing through task stealing
- Task distribution schemes

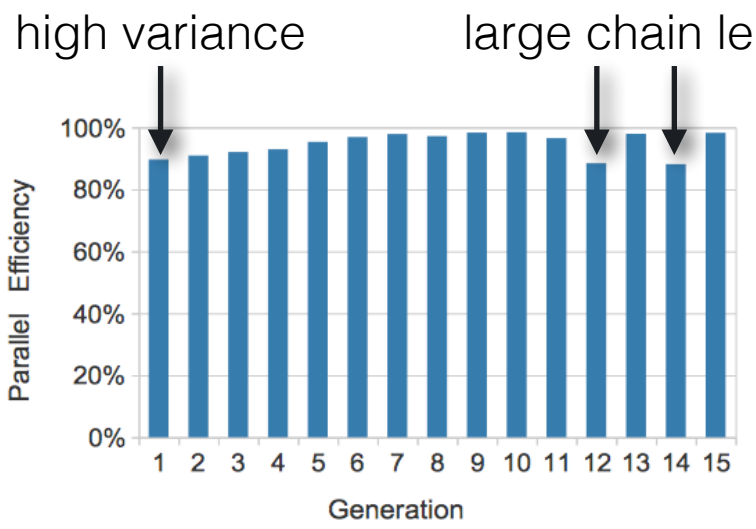
Test Case 1: TIP5 water model

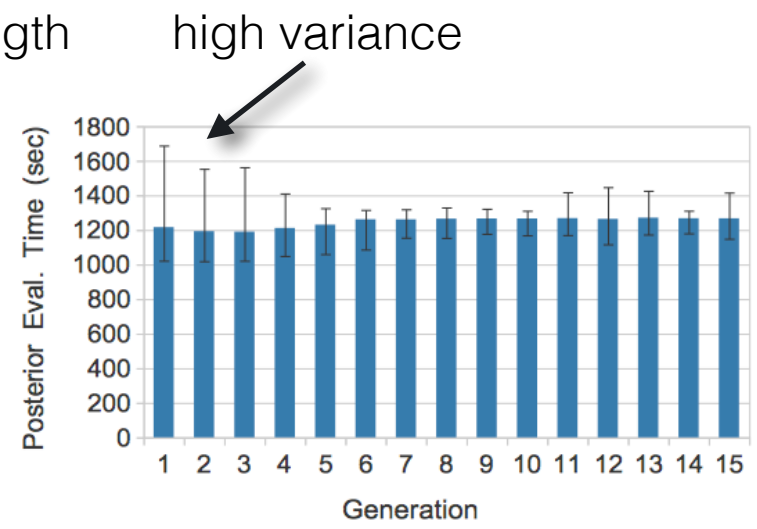
- Calibration of the most widely used Molecular Dynamics (MD) model
  - Using TMCMC and CMA-ES
- Target hardware platform: Piz Daint (Cray XC30) as CSCS
  - Each node: 8-core Intel Xeon E5-2670 + 1 NVIDIA K20x GPU
- A single function evaluation includes
  - 2 MD simulations using GROMACS on CPU+GPU
  - a Matlab-based post-processing stage



Performance

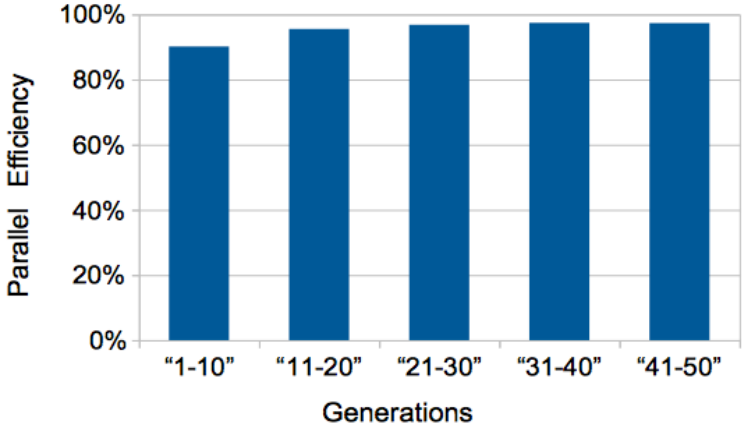
TMCMC



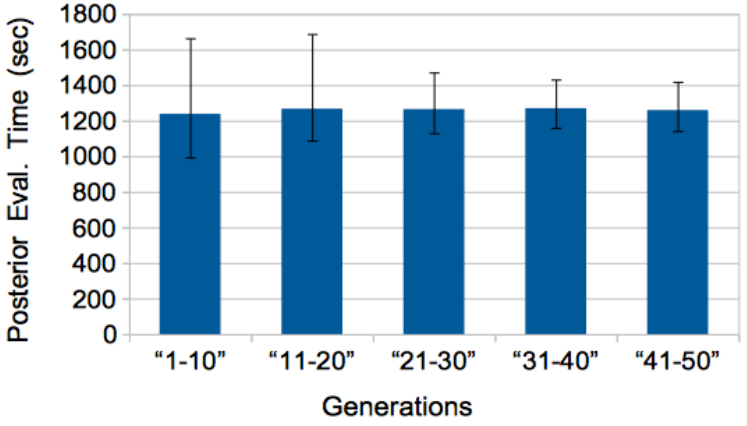


CMA-ES

Efficiency



Posterior Eval. Time



Test Case 2: LJ Potential

- Calibration of the Lennard-Jones potential parameters for Helium
  - Using the ABC-SubSim algorithm
- Function evaluation:
  - MD simulation using the MPI parallel version of LAMMPS
- 512 nodes of Piz Daint:
  - 4096 cores in total, 2 TORC workers/node
- #samples=15360, chain length=5
  - 3 chain tasks per TORC worker

Level	T_f	T_w	Efficiency
0	82 ± 83	1497	81.8%
1	87 ± 57	1843	70.5%
2	68 ± 10	1237	81.9%
3	65 ± 6	1110	88.4%
4	66 ± 5	1078	92.2%

Software Installation

- Configure and Install
  $ ./configure CC=mpicc --prefix=${HOME}/usr/torc --with-maxvps=48
  $ make
  $ make install

- ${HOME}/usr/torc/bin should be added in $PATH
- torc_flags and torc_libs are 2 useful scripts for easy compilation of applications

- $ torc_cflags
  -I/cluster/home/mavt/chatzidp/usr/torc/include -D_REENTRANT
- $ torc_libs
  -L/cluster/home/mavt/chatzidp/usr/torc/lib -ltorc –lpthread

Compilation

```
$ cat Makefile

MPICC=mpic++
CFLAGS=-O3 -pthread -std=c++11 -fpermissive
CFLAGS+=`torc_cflags`
LIBS=`torc_libs`

all: simpson_torc

simpson_torc: simpson_torc.cpp
   $(MPICC) $(CFLAGS) -o simpson_torc simpson_torc.cpp $(LIBS)
```

Brutus Node, 8 cores, MPICH2

1 process, 1 worker/process

```
[chatzidp@a6096 examples_torc]$ time mpirun -env TORC_WORKERS 1  -n 1 ./
simpson_torc
7.84669

real    0m11.683s
user    0m11.669s
sys     0m0.010s
```

1 process, 8 workers/process

```
[chatzidp@a6096 examples_torc]$ time mpirun -env TORC_WORKERS 8 -n 1 ./
simpson_torc
7.84669

real    0m1.730s
user    0m11.416s
sys     0m0.060s
```

Brutus Node, 8 cores, MPICH2

8 processes, 1 worker/process

```
[chatzidp@a6096 examples_torc]$ time mpirun -env TORC_WORKERS 1  -n 8 ./
simpson_torc
7.84669

real    0m1.685s
user    0m11.329s
sys     0m0.045s
```

4 processes, 2 workers/process

```
[chatzidp@a6096 examples_torc]$ time mpirun -env TORC_WORKERS 4  -n 2  ./
simpson_torc
7.84669

real    0m1.693s
user    0m11.342s
sys     0m0.029s
```