An HPC Framework for Bayesian Uncertainty Quantification

Panagiotis Hadjidoukas

Chair for Computational Science
ETH Zurich

with: all members of the CSE Lab

CSE**lab**
Computational Science & Engineering Laboratory
http://www.cse-lab.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Uncertainty Quantification

- Simulations imply models, their numerical discretization and their deployments on computers

- Errors: observational, modeling, numerical, computing

- Bayesian Uncertainty Quantification and Prediction (UQ+P): effective integration of Data with Simulation models

- Aggregate scientific knowledge obtained by ensembles of simulation runs

Bayesian UQ: Model Selection and Calibration

Experimental Data: D

Use observations to select the model classes and estimate their parameter values such that the model predictions best fit the data

PARAMETER ESTIMATION

$$f(\theta_i|D,\mathcal{MD}_i) = \frac{f(D|\theta_i,\mathcal{MD}_i)\,\pi(\theta_i|\mathcal{MD}_i)}{f(D|\mathcal{MD}_i)}$$

Experiments

Physical limitations
Past studies
Expert elicitation

MODEL CLASS SELECTION

$$Pr(\mathcal{MD}_i|D) = \frac{f(D|\mathcal{MD}_i)\,Pr(\mathcal{MD}_i)}{f(D)}$$

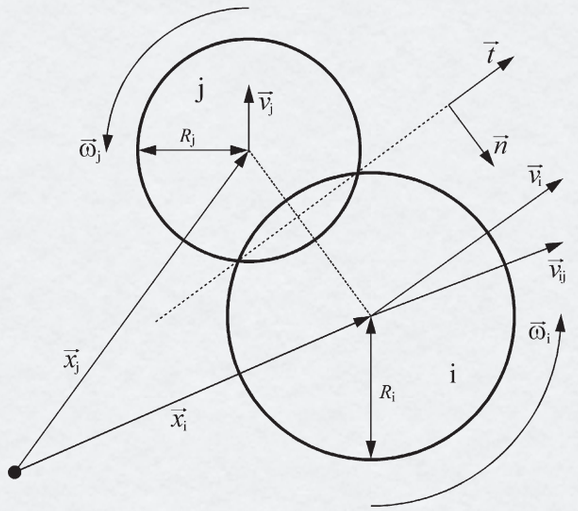
Evidence of Model Class

$$f(D|\mathcal{MD}_i) = \int f(D|\theta_i,\mathcal{MD}_i)\,\pi(\theta_i|\mathcal{MD}_i)\,d\theta_i$$

Example: Discrete Element Method

Models

Particle-particle interactions



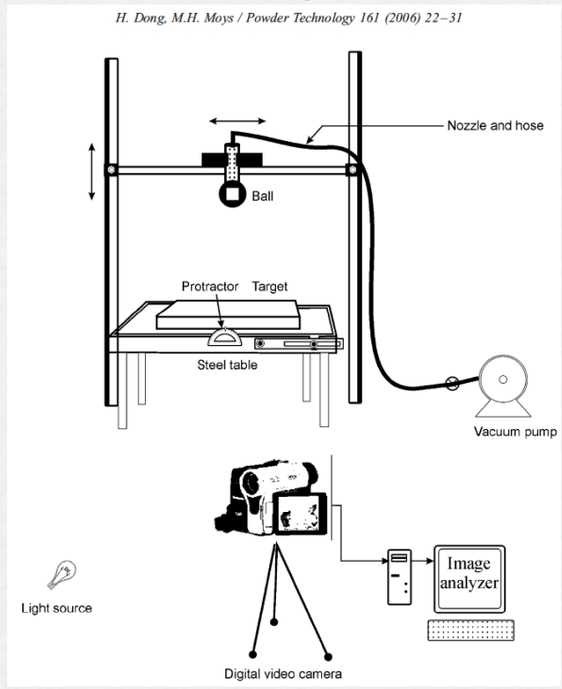
$$\mathbf{F}^n = -k^n \mathbf{y} - \gamma^n \frac{d\mathbf{y}}{dt} |\mathbf{y}|^{\Theta},$$

$$\mathbf{F}^t = \min\left(\frac{2}{3} k^s \boldsymbol{\xi}^t, \mu_f \mathbf{F}^n\right)$$

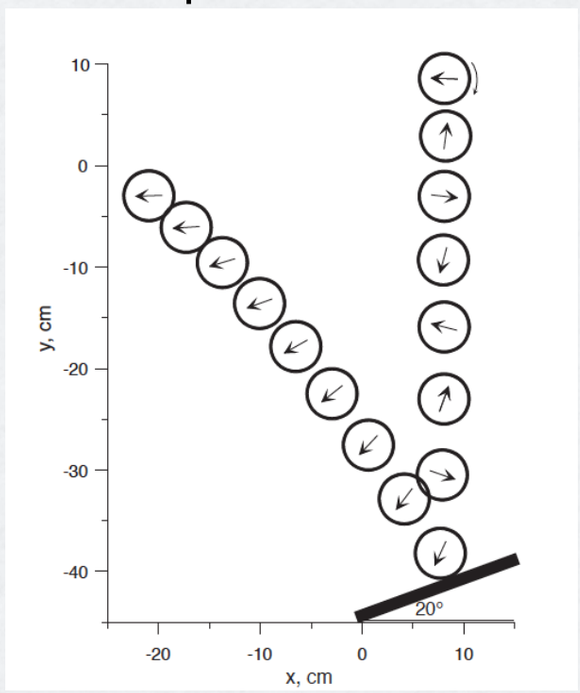
$$\boldsymbol{\xi}^t = \int_{t_0}^{t} \boldsymbol{v}_{rel}^t(\tau)\, d\tau,$$

$$k^n = \frac{4}{3} E_{eff} \sqrt{R_{eff} |\mathbf{y}|},$$

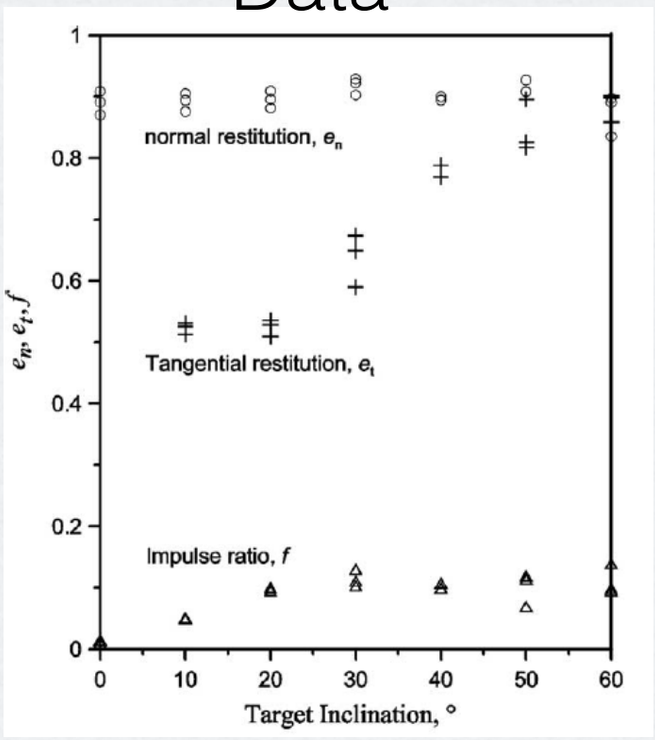
Setup



Experiment



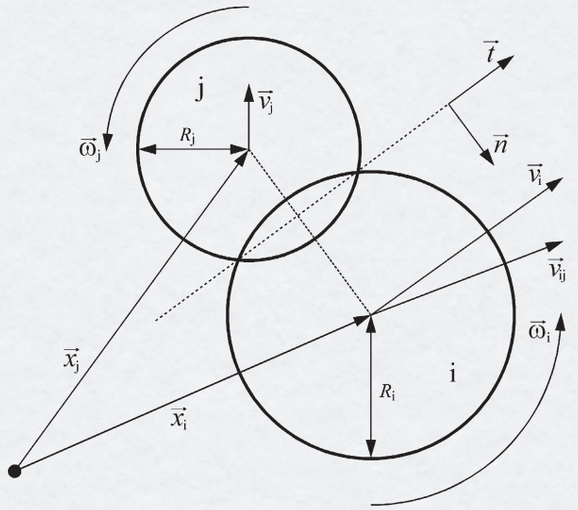
Data



Hadjidoukas, P.E., Angelikopoulos, P., Rossinelli, D., Alexeev, D., Papadimitriou, C. and Koumoutsakos, P. (2014) "Bayesian Uncertainty Quantification and propagation for Discrete Element Simulations of granular materials", *Computer Methods in Applied Mechanics and Engineering*

Example: Discrete Element Method

Particle-particle interactions



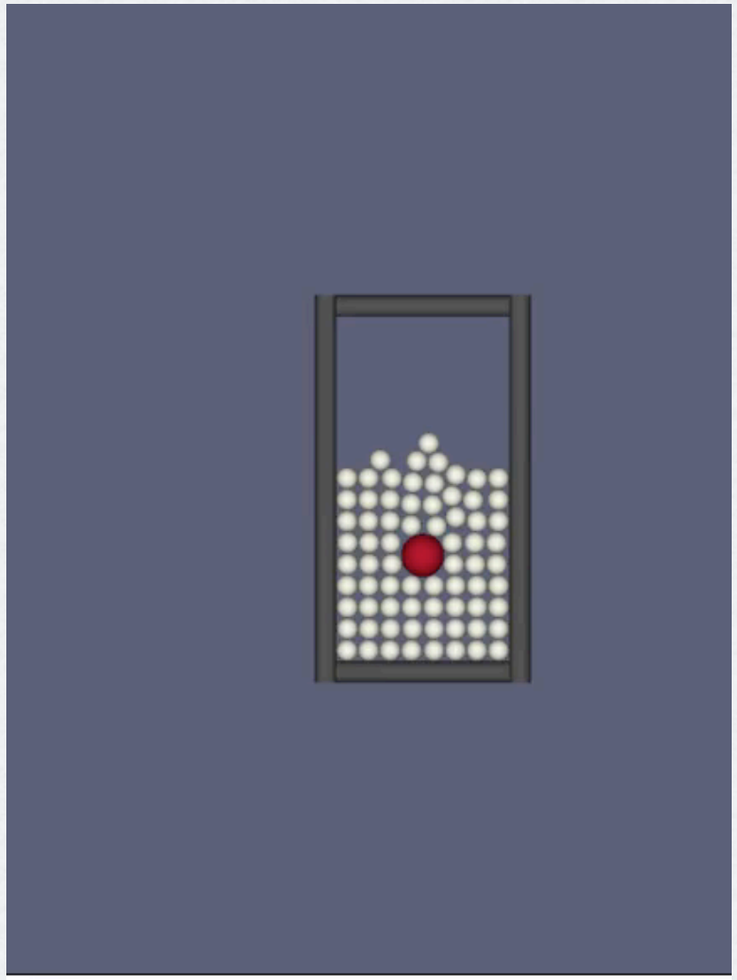
$$\mathbf{F}^n = -k^n \mathbf{y} - \gamma^n \frac{d\mathbf{y}}{dt} |\mathbf{y}|^{\Theta},$$

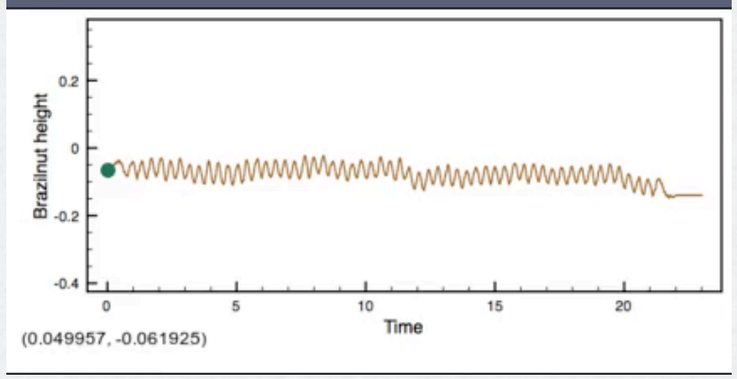
$$\mathbf{F}^t = \min\left(\frac{2}{3} k^s \boldsymbol{\xi}^t, \mu \mathbf{F}^n\right)$$

$$\boldsymbol{\xi}^t = \int_{t_0}^{t} \boldsymbol{v}_{rel}^t(\tau)\, d\tau,$$

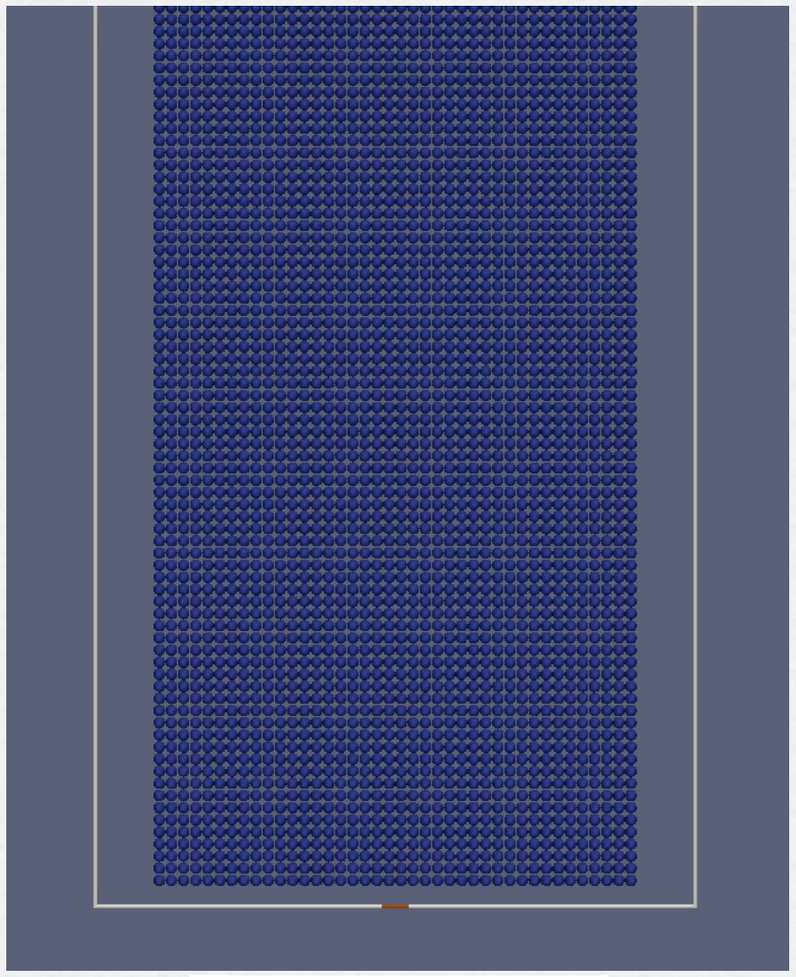
$$k^n = \frac{4}{3} E_{eff} \sqrt{R_{eff} |\mathbf{y}|},$$

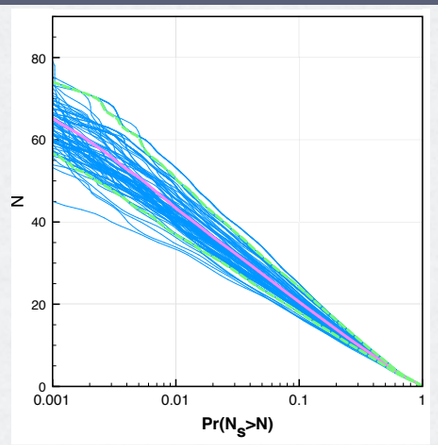
Brazil Nut effect





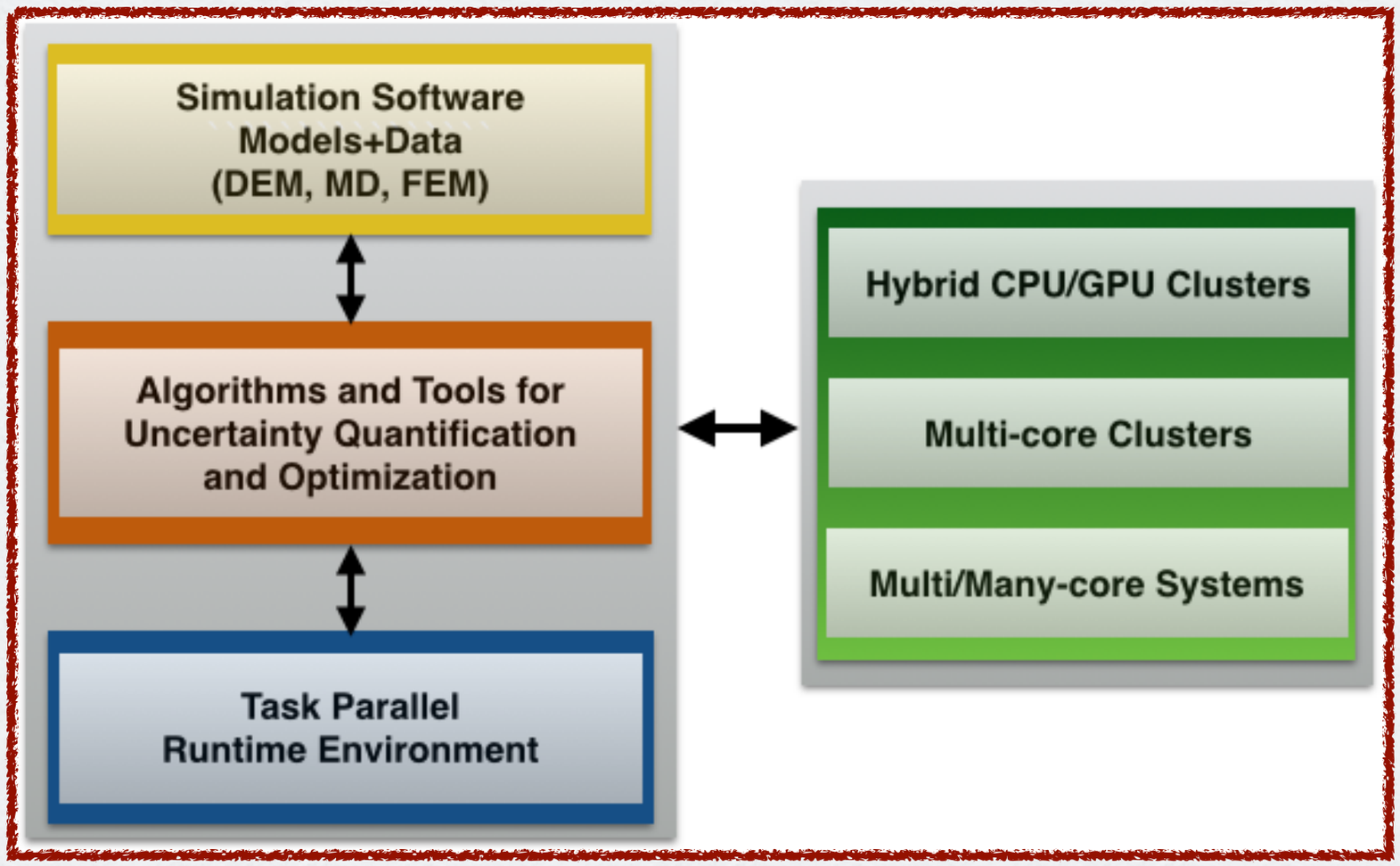
Silo Discharge





Π4U: HPC Framework for Bayesian UQ

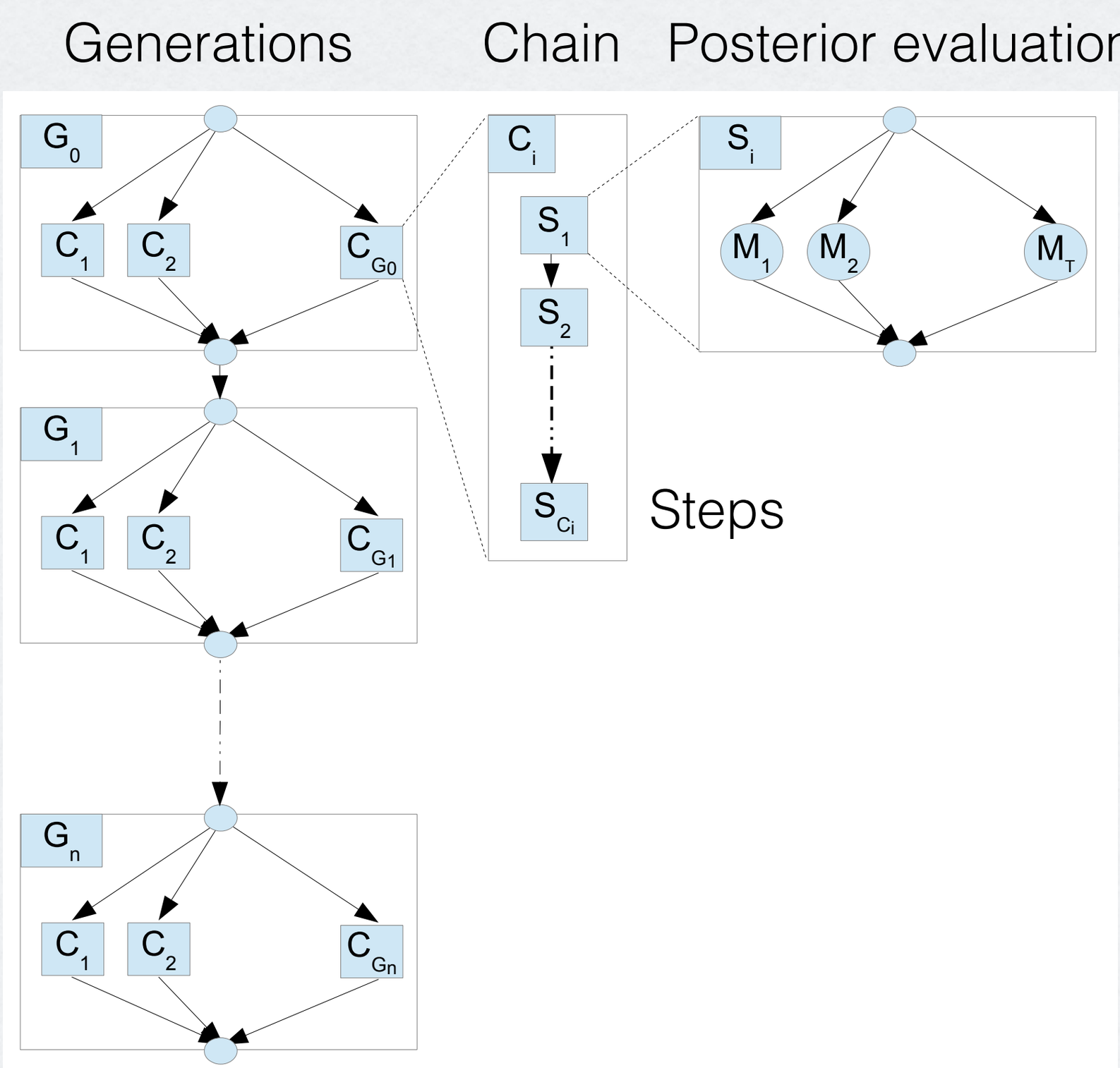
TMCMC
ABC-Subsim
CMAES
…



- ✓ Platform agnostic task-based parallelism
- ✓ Multi-level parallelism
- ✓ Transparent load balancing
- ✓ Extensible/ built upon the TORC tasking library

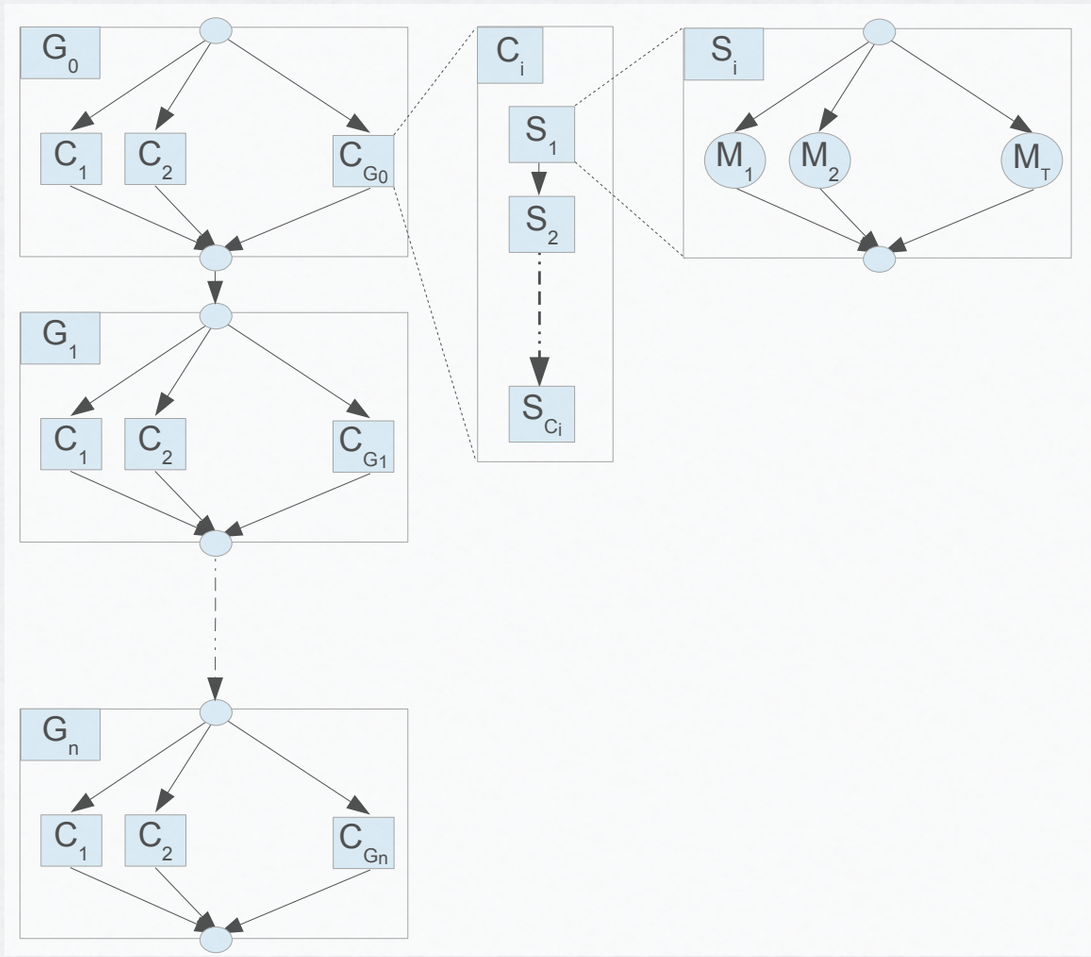
Hadjidoukas, P.E., Angelikopoulos, P., Papadimitriou, C. and Koumoutsakos, P. (2015) "Pi4U: A high performance computing framework for Bayesian uncertainty quantification of complex models", *Journal of Computational Physics*

Transitional MCMC: task-graph



HPC Challenges

- **Simulations**
  - variable computational cost
  - can generate large volume of data
- **Algorithms**
  - include multiple levels of parallelism
  - introduce load imbalance
- **Hardware**
  - variety of computing resources, highly heterogeneous
- **Programming methodology**



Programming Approach - OpenMP

```
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
    {
       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
    }

   return 0;
}
```

Programming Approach - TORC

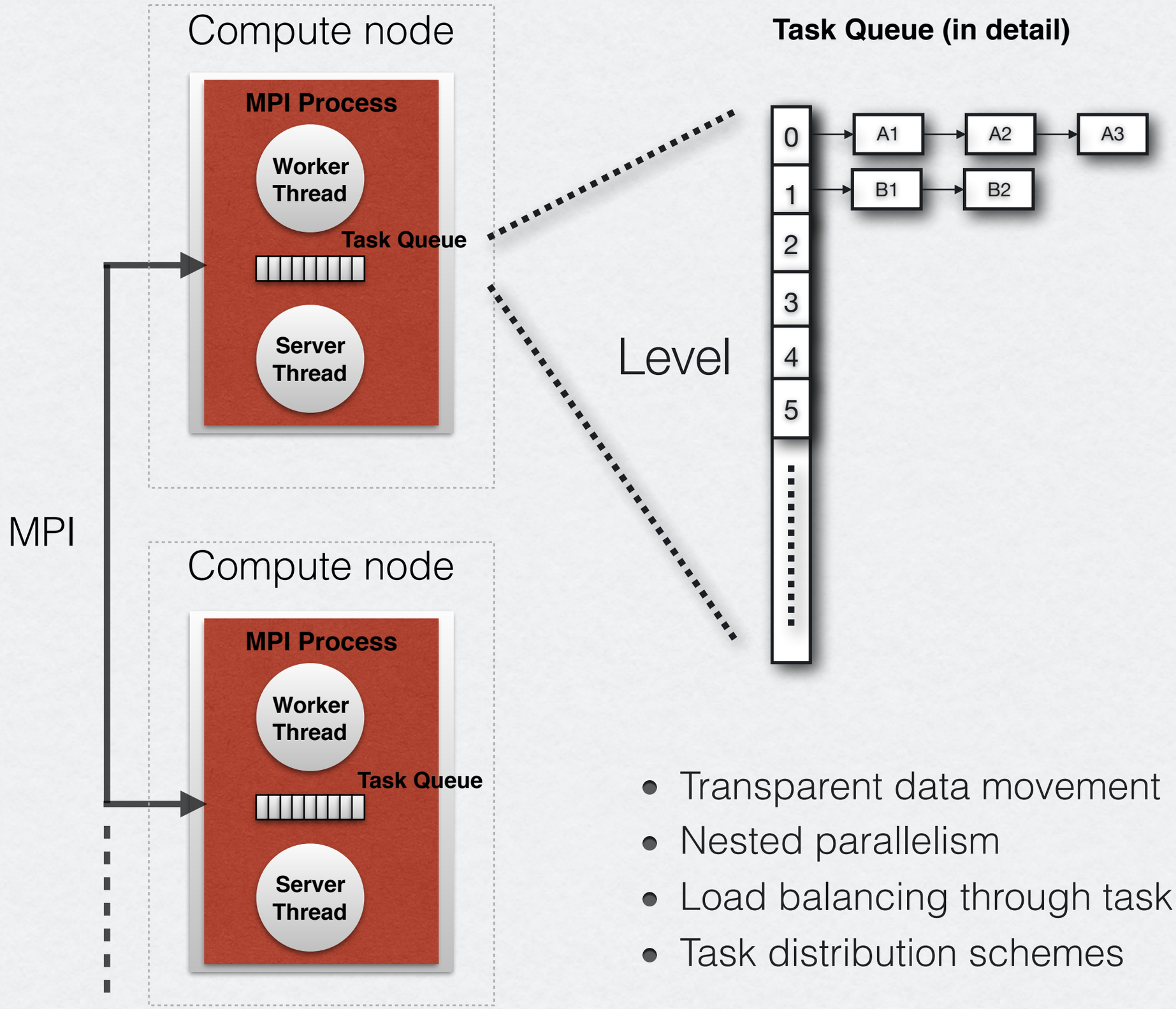
```
void task(double *x, double *y) { *y = x[0] + x[1]; }

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 sequential/parallel code
- call an external application
- submit a job

Runtime Architecture



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

Test cases

- UQ (and Optimization) for:
  - Molecular Dynamics (MD): GROMACS and LAMMPs
  - Dissipative Particle Dynamics (DPD): uDeviceX
  - Finite Volume Flow Solvers: CUBISM-MPCF

Test Case 1: TIP5-E water model

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