On building parallel algorithms and software for hydraulic tomography

Xing Cai

Simula Research Laboratory, Norway

Department of Informatics, University of Oslo

SIAM-GS07, March 22, 2007

Outline

Acknowledgements

- Partially supported by the Norwegian Research Council
- Collaborators:
  - Jim Yeh
  - Junfeng Zhu

From a computational viewpoint

- Mathematically, *hydraulic tomography* (HT) is an inverse problem involving partial differential equations (PDEs)
- High resolution $\rightarrow$ large computational amount $\rightarrow$ parallelization is imperative
- Parallel solution of PDEs: a well studied topic, but not widely used in HT
- Objective: incorporation of parallel computing into HT

List of Topics

Hydraulic tomography (HT)

A high-resolution technique for characterizing the heterogeneous distribution of hydraulic quantities in aquifers

Find the hydraulic conductivity $K(\mathbf{x})$ and specific storage $S_s(\mathbf{x})$ that fit with

$$\nabla \cdot [K(\mathbf{x})\nabla H] + Q(\mathbf{x}_p) = S_s(\mathbf{x})\frac{\partial H}{\partial t} \tag{1}$$

subject to measurements of H and suitable boundary and initial conditions

Basic strategy of HT

- A series of pumping tests
- Excitation and response
- Values of H are measured at a set of locations $\mathbf{x}_j$ $(1 \leq j \leq n_h)$ and for a set of time levels t_ℓ $(1 \leq \ell \leq n_t)$
- An inverse problem with respect to K and S_s

Basic geostatistical assumptions

- Assume $\ln K = \bar{K} + f$ and $\ln S_s = \bar{S} + s$, i.e., mean value plus perturbation
- Assume $H = \bar{H} + h$
- $$\nabla \cdot \left[\bar{K}_{\mathrm{con}}(\mathbf{x}) \nabla \bar{H}_{\mathrm{con}}\right] + Q(\mathbf{x}_p) = \bar{S}_{\mathrm{con}}(\mathbf{x}) \frac{\partial \bar{H}_{\mathrm{con}}}{\partial t}$$

  where $\bar{K}_{\mathrm{con}}$, $\bar{S}_{\mathrm{con}}$ and $\bar{H}_{\mathrm{con}}$ represent *conditional effective* hydraulic conductivity, specific storage and hydraulic head

Example: SSLE

- Sequential succesive linear estimator (SSLE) is one possible solution scheme for HT
  - developed by Yeh and Liu
  - a geostatistical approach for (transient) HT
- Primary variables: perturbations $f(\mathbf{x})$ and $s(\mathbf{x})$
- Secondary varaible: perturbation $h(\mathbf{x}, t)$
-

$$H(\mathbf{x}, t) = \bar{H}(\mathbf{x}, t) + f(\mathbf{x}) \left.\frac{\partial H(\mathbf{x}, t)}{\partial \ln K(\mathbf{x})}\right|_{\bar{K},\bar{S}} + s(\mathbf{x}) \left.\frac{\partial H(\mathbf{x}, t)}{\partial \ln S_s(\mathbf{x})}\right|_{\bar{K},\bar{S}}$$

- Sensitivity terms $\frac{\partial H(\mathbf{x},t)}{\partial \ln K(\mathbf{x})}$ and $\frac{\partial H(\mathbf{x},t)}{\partial \ln S_s(\mathbf{x})}$ are updated gradually
- Covariance of h, cross-covariance between h and f, between h and s are also updated gradually by cokriging

More on SSLE

- To update the sensitivity terms $\frac{\partial H(\mathbf{x},t)}{\partial \ln K(\mathbf{x})}$ and $\frac{\partial H(\mathbf{x},t)}{\partial \ln S_s(\mathbf{x})}$:
  - For each pair of $(\mathbf{x}_j, t_\ell)$, solution of an adjoint backward transient problem:

$$S_s \frac{\partial \phi^*}{\partial t} + \nabla \cdot (K \nabla \phi^*) = \delta(\mathbf{x} - \mathbf{x}_j)(t - t_\ell), \tag{2}$$

  - Calculation of the sensitivity terms:

$$\frac{\partial H(\mathbf{x}_j, t_\ell)}{\partial \ln K(\mathbf{x}_i)} = \int_T \int_\Omega \left\{ \frac{\partial K(\mathbf{x})}{\partial \ln K(\mathbf{x}_i)} \nabla \phi^* \cdot \nabla H(\mathbf{x}_j, t_\ell) \right\} d\mathbf{x} dt, \tag{3}$$

$$\frac{\partial H(\mathbf{x}_j, t_\ell)}{\partial \ln S_s(\mathbf{x}_i)} = \int_T \int_\Omega \left\{ \frac{\partial S_s(\mathbf{x})}{\partial \ln S_s(\mathbf{x}_i)} \phi^* \frac{\partial H(\mathbf{x}_j, t_\ell)}{\partial t} \right\} d\mathbf{x} dt. \tag{4}$$

List of Topics

We need parallel computing

- The aquifer domain Ω may be large
- High resolution of K and S_s may be desired
- Total amount of computation grows fast with N (number of mesh points in Ω)
- If $N \sim \mathcal{O}(10^6)$ or more, parallelization is imperative

Identifying different sources of parallelism

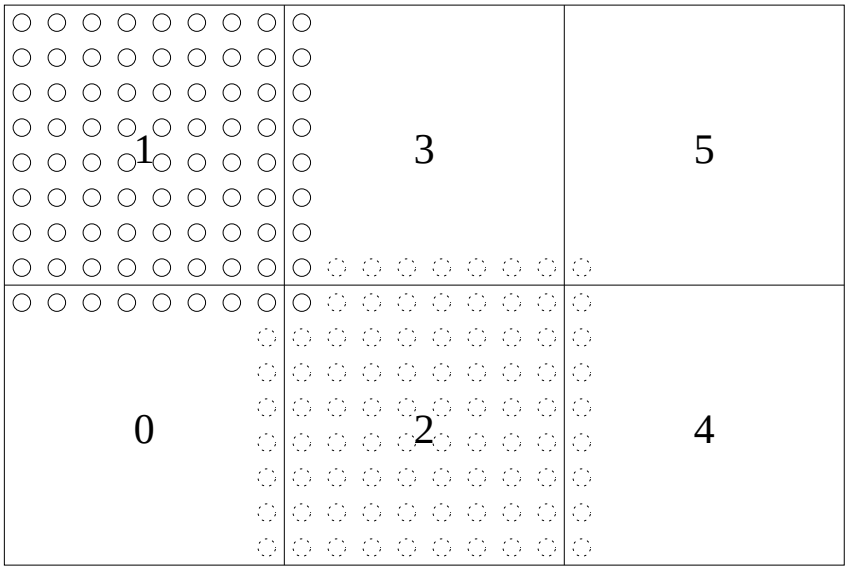
Using SSLE as an example for the solution scheme

- The adjoint state equation $S_s \frac{\partial \phi^*}{\partial t} + \nabla \cdot (K \nabla \phi^*) = \delta(\mathbf{x} - \mathbf{x}_j)(t - t_\ell)$ needs to be solved for each pair of $(\mathbf{x}_j, t_\ell)$ — *task parallelism*
- The sensitivity terms, covariance of h, cross-covariance between h and f, between h and s, are calculated for all the elements in Ω, independent of each other — *data parallelism*
- Parallelism can also be obtained within one solve of the forward problem $\nabla \cdot [K(\mathbf{x}) \nabla H] + Q(\mathbf{x}_p) = S_s(\mathbf{x}) \frac{\partial H}{\partial t}$ — *subdomain-based parallelization*

Handling different sources of parallelism

- Task parallelism is straightforward, when the number of $(\mathbf{x}_j, t_\ell)$ pairs is much larger than the number of processors
- Data parallelism is straightforward, when the number of elements is divided evenly among the processors
- Subdomain-based parallelization requires some effort:
  - Solving the forward problem involving all the processors
  - Need to divide the work in the discretization phase
  - Need to divide the work in the linear solution phase
  - The processors are more tightly coupled

Subdomain-based parallelization



- Ω is decomposed into $\{\Omega_s\}$, $1 \leq s \leq P$
- Partitioning of domain $\Rightarrow$ division of work
- Subdomain Ω_s is the "responsibility of processor s
- In such a parallelized PDE solver:
  - Subdomains do serial computations most of the time
  - Collaboration between subdomains is via communication

More on subdomain-based parallelization

- Work division arises from domain decomposition
- Easy parallelization of the discretization phase
  - The global system $\mathbf{Ax} = \mathbf{b}$ is naturally distributed
  - No communication is needed in the discretization phase
- Collaborative work is needed when solving $\mathbf{Ax} = \mathbf{b}$
  - Serial linear algebra operations on each processor
  - Inter-processor commuication is needed

Solving linear systems

$$\mathbf{Ax} = \mathbf{b}$$

- Most time-consuming computation at each time step when solving (1) and (2)
- Can be parallelized by the subdomain-based approach
  - Each subdomain *independently* forms $\mathbf{A}_s$, $\mathbf{x}_s$, and $\mathbf{b}_s$
  - Parallel solution of the *distributed* global system $\mathbf{Ax} = \mathbf{b}$

Parallel solvers of $\mathbf{Ax} = \mathbf{b}$

- Iterative solvers are best suited for parallelization
- Example: *conjugate gradient* method

  Initially: $r = b - Ax$, $p = r$, $\pi^0_{r,r} = (r, r)$

  Iterations:

$w = Ap$	*matrix-vector product*
$\pi_{p,w} = (p, w)$	*inner product*
$\xi = \pi^0_{r,r}/\pi_{p,w}$	
$x = x + \xi p$	*vector addition*
$r = r - \xi w$	*vector addition*
$\pi^1_{r,r} = (r, r)$	*inner product*
$\beta = \pi^1_{r,r}/\pi^0_{r,r}$	
$p = r + \beta p$	*vector addition*
$\pi^0_{r,r} = \pi^1_{r,r}$	

Parallel linear algebra kernels

- Vector addition: $\mathbf{w} = \mathbf{u} + \mathbf{v}$
  - On subdomain Ω_s: $\mathbf{w}_s = \mathbf{u}_s + \mathbf{v}_s$
  - No communication is needed
- Inner product: $c = \mathbf{u} \cdot \mathbf{v} = \sum u_i \, v_i$
  - On subdomain Ω_s: $c_s = \mathbf{u}_s \cdot \mathbf{v}_s = \sum u_{s,i} v_{s,i}$
  - All-to-all communication: $c = c_1 + c_2 + \ldots + c_P$
- Matrix-vector product: $\mathbf{v} = \mathbf{A}\mathbf{u}$
  - On subdomain Ω_s: $\mathbf{v}_s = \mathbf{A}_s \mathbf{u}_s$
  - One-to-one communication between each pair of neighbors

List of Topics

Test case of parallel HT

Starting point: serial Fortran HT code from Univ. Arizona

- Straightforward parallelization of (2), (3)-(4), plus insertion of a few MPI commands (e.g. `MPI_Allgatherv`)
- Subdomain-based parallelization of (1) using Diffpack (C++ PDE libraries, see `www.diffpack.com`)

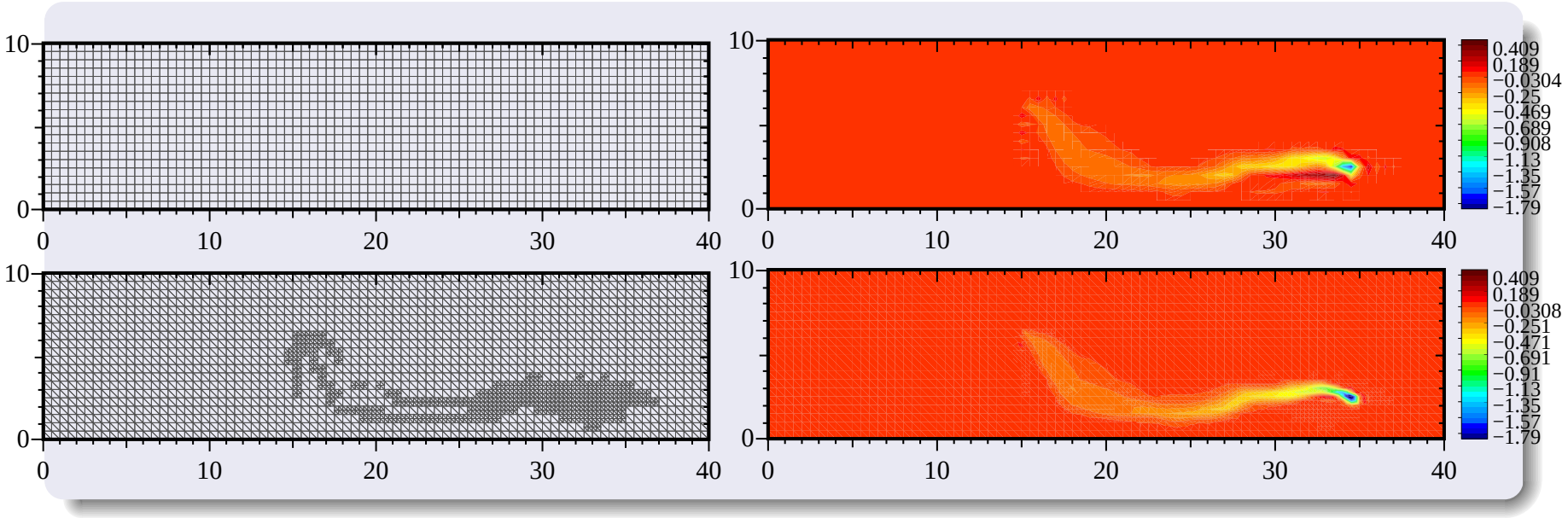
Preliminary CPU measurements

A test 3D case: $45 \times 45 \times 45$ mesh

- Linux cluster: 1.3 GHz Itanium II processors + Gbit ethernet

Processors	CPU (minutes)	Speedup
1	29.55	N/A
2	15.45	1.91
4	7.89	3.75
8	4.07	7.26
16	1.81	16.33
24	1.27	23.27

Parallel adaptive computing



- Adaptive mesh refinement
- Each subdomain adds roughly same amount of new points
- Non-matching subdomain meshes maybe between neighbors

Remarks

- Subdomain-based parallelization is well-suited for PDEs, also for HT
- Development of parallel HT software can be a challenge:
  - legacy codes are useful, but with lots of low-level details
  - need a framework to ease the parallelization work
- Perfect speedups are not realistic:
  - some parts of the code (e.g. I/O) are not suitable for parallelization
  - commuication overhead is inevitable
  - some level of load imbalance
  - some duplicated computations
- However, reasonably good parallel performance is achievable!