(Portable) Task-based programming model for elastodynamics

Hoscar workshop - Sophia Antipolis, September 21-24 2015

Julien Diaz – INRIA Magique3D, Pau

- Hélène BARUCQ, Lionel BOILLOT – INRIA Magique3D, Pau
- Henri CALANDRA – TOTAL E&P, Houston
- Emmanuel AGULLO – INRIA Hiepacs, Bordeaux
- George BOSILCA – ICL, University of Tennessee
Heterogeneity is everywhere

Figure: “Welcome to the jungle”, Herb SUTTER, 2012

- Architectures: Xeon, Cell, MIC, KALRAY, GPU, FPGA, ...
- Memories: RAM, caches, (cc)NUMA, ...
- Programmation: MPI, CUDA/OpenCL, OpenACC, ...

⇒ need of efficient portable programming model, easy to implement!
# Table of contents

1. **Elastic wave simulation**
   - Wave equation algorithm
   - Parallel implementation issue

2. **Task-based paradigm**
   - Task-based formulation

3. **Numerical results**
   - Experimental setup
   - ccNUMA machine
   - Intel Xeon Phi
Table of contents

1. Elastic wave simulation
   - Wave equation algorithm
   - Parallel implementation issue

2. Task-based paradigm
   - Task-based formulation

3. Numerical results
   - Experimental setup
   - ccNUMA machine
   - Intel Xeon Phi
Discretization

Elastic wave equation (first order)

\[
\begin{align*}
\rho(x) \partial_t \mathbf{v}(x, t) &= \nabla \cdot \mathbf{\sigma}(x, t) \\
\partial_t \mathbf{\sigma}(x, t) &= C(x) : \varepsilon(\mathbf{v}(x, t)) \tag{1}
\end{align*}
\]

Discontinuous Galerkin with Leap-frog scheme

Iteration on \( n \):

\[
\begin{align*}
M_v \frac{\mathbf{v}_h^{n+1} - \mathbf{v}_h^n}{\Delta t} + R_{\mathbf{\sigma}} \mathbf{\sigma}_h^{n+1/2} &= 0 \\
M_{\mathbf{\sigma}} \frac{\mathbf{\sigma}_h^{n+3/2} - \mathbf{\sigma}_h^{n+1/2}}{\Delta t} + R_{\mathbf{v}} \mathbf{v}_h^{n+1} &= 0 \tag{2}
\end{align*}
\]

\( M_v \) and \( M_{\mathbf{\sigma}} \) block-diagonal matrices \( \Rightarrow \) easily invertible!
Parallel algorithm

**Algorithm 1: DIP parallel**

**Data:** $N_p, N_h, \Delta t, N_t$

**Result:** $v_h, \sigma_h$

$N_{h_{loc}} \leftarrow \text{DomainDecomposition}(N_p);$

$[v_h^1, \sigma_h^{3/2}] \leftarrow \text{Initialization}(N_{h_{loc}}, \Delta t);$

**for** $n = 1..N_t$ **do**

\[ \sigma_{h_K}^{n+1/2} \leftarrow \text{Communication}(\sigma_{h_K}^{n+1/2}); \]

**for** $K = 1..N_{h_{loc}}$ **do**

\[ v_{h_K}^{n+1} \leftarrow \text{UpdateVelocity}(v_{h_K}^n, \sigma_{h_K}^{n+1/2}, \Delta t); \]

**end**

\[ v_{h_K}^{n+1} \leftarrow \text{Communication}(v_{h_K}^{n+1}); \]

**for** $K = 1..N_{h_{loc}}$ **do**

\[ \sigma_{h_K}^{n+3/2} \leftarrow \text{UpdateStress}(\sigma_{h_K}^{n+1/2}, v_{h_K}^{n+1}, \Delta t); \]

**end**

**end**
MPI version = one domain per process

Figure: Domain decomposition example

Load balancing limitations:
- order of (discretization of) each cell
- order of the neighbor cells (for the face terms)
- architecture specifications (e.g. SIMD, cache size, ... )
Load imbalance example

Figure: Trace analysis of 10 timesteps on 32 MPI processes

Each line represents a core activity during the execution time:
- **ORANGE** – computation of the velocity field
- **RED** – computation of the stress tensor
- **GREY** – waiting time (synchronization)
Table of contents

1 Elastic wave simulation
   - Wave equation algorithm
   - Parallel implementation issue

2 Task-based paradigm
   - Task-based formulation

3 Numerical results
   - Experimental setup
   - ccNUMA machine
   - Intel Xeon Phi

(INRIA – TOTAL)
Task-based programming

Principles
- Determine dependencies between tasks with the Bernstein’s conditions
- Specify the data modes: Read/Write

⇒ creation of a DAG (Direct Acyclic Graph)

\begin{align*}
\text{fun1}(A: \text{inout}, B: \text{out}) \\
\text{fun2}(B: \text{in}, C: \text{out}) \\
\text{fun3}(A: \text{inout}, C: \text{in})
\end{align*}

Original control-flow becomes useless (!) and an obstacle to efficiency
DAG of DIP

Figure: One iteration of DIP on three sub-domains
Fine granularity

Figure: Subdivision example

More than one domain per CPU

- exhibit deeper parallelism
- allow dynamic flexibility
- reduce the boundary size
1 Elastic wave simulation
   • Wave equation algorithm
   • Parallel implementation issue

2 Task-based paradigm
   • Task-based formulation

3 Numerical results
   • Experimental setup
   • ccNUMA machine
   • Intel Xeon Phi
Geophysics test case

Realistic test case:
- 3D elastic
- TTI (anisotropy)
- multi-layers

Hybrid discretization:
- unstructured tetrahedra
- P1-P2-P3 orders
- boundary conditions
ccNUMA machine

- 6 processors (previous Intel Xeon E7-8837)
- Total of 48 CPU cores in ccNUMA architecture

Figure: cache-coherent Non-Uniform Memory Access (ccNUMA) scheme
ccNUMA results - speedup

Figure: With multi-virtual processes

⇒ virtual processes are key-point for NUMA-awareness

Figure: With no virtual process
Numerical results

ccNUMA machine

Trace comparison

**Figure**: MPI-based $t = 2.517s$

**Figure**: PaRSEC version (multi-VP, granularity x6) $t = 2.060s$
Intel Xeon Phi configuration

General specifications:
- AVX instruction set
- hyperthreading by 4
- no L3 cache memory
- shared L2 (ring bus)

Co-processor 7120P:
- 61 CPUs at 1.238 GHz (244 threads)
- 64 KB of L1 cache (per core)
- 512 KB of L2 cache (per core)
- 16 GB of DRAM (scrappy)
On the 60 cores:
- **very good speedup**
- **fine** granularity
- **good** efficiency

Hyperthreading enabled:
- **quite good speedup**
- **coarse** granularity
- **bad** efficiency

Idea → use hyperthreading to manage load/store
Conclusion and perspectives

Context
- compute & exchange algorithm
- domain decomposition parallelism

Key-points
- fine granularity and work-stealing
- virtual process capability of PaRSEC

Results
Shared memory architectures (ccNUMA machine, Xeon Phi co-processor) ⇒ Next coming: distributed memory machines

Support by INRIA-TOTAL strategic action DIP (http://dip.inria.fr)