An Efficient Pipelined Implementation of Space-Time Parallel Applications

Toshiya Takami and Daiki Fukudome
Kyushu University

Sep. 10-13, 2013 ParCo’13, Munich

Outline

• Introduction of the standard Parareal-in-time method
  • Limitations of the standard implementation
• New Implementation by the use of an Identity Operator
  • iPareal: Identity Parareal for time evolution problems
  • Convergence property and timelines in applications
• Bucket Brigade Communications and Optimization
  • Overlapping communications and computations
  • Pipelined optimization and its speed-up
• Summary & conclusion

1. Introduction

Parareal-in-Time Method

• Parallelization of strictly dependent calculations:
  • is widely applied to time-domain decomposition problems
  • is applicable to general iterative computations
• Instead of the exact Sequence: \( x_{k+1} = F_k(x_k) \)
• \((r+1)\)-th Approximate Sequence:
  \[ x_{k+1}^{(r+1)} = G_k(x_k^{(r+1)}) + F_k(x_k^{(r)}) - G_k(x_k^{(r)}) \]
  • \( G_k(x) \) is a coarse solver, which should be defined by users.
  • For \( r \to \infty \), \( x_k^{(r)} \) converges to the exact sequence \( x_k \).
Large-scale Applications of the Parareal-in-Time Method

• Many-Task Computing:
  • Workshop in SCI11 (Seattle): MTAGS
  • Technical Paper in SC12 (Salt Lake City)

• Space-Time Parallelism:
  • A massive space-time parallel N-body solver

1. Introduction

Speedup Ratio by the Standard Parareal-in-Time Method

• Speedup Ratio:
  \[ S = \frac{T_f}{T_g + T_c + \frac{R(T_f + T_c)}{K}} \]
  \( T_c, T_f, T_g \): costs of communication, and computation \( T_k(x), G_k(x) \)
  \( P \): the number of resources used for the time direction
  \( R \): the order of approximation
  \( K \): length of the series (= P+R-1)

1. Introduction

Limitations of the Standard Parareal-in-Time Method

• A tailor-made coarse solver \( G_k(x) \) is necessary:
  • \( G_k(x) \) must be faster than \( F_k(x) \) and a good approximation.
  • The final speedup is bounded by \( T_f / (T_g + T_f) \).

• The program is relatively complicated because of dependent communications between processes:
  • difficulty in fine tuning of the communications

We introduce ‘iParareal’:
Identity operator is used as the coarse solver

1. Introduction

iParareal Algorithm

• An identity operator (doing nothing) is used instead of the coarse solver \( G_k(x) \).

  iParareal iteration formula:
  \[ x_{k+1}^{(r+1)} = x_k^{(r+1)} + F_k(x_k^{(r)}) - x_k^{(r)} \]

• Note that convergence is limited to the following case:
  • Continuous time evolutions represented by a small \( dt \)
  \[ x(t_k + dt) = F(x(t_k)) \iff x(t_{k+1}) = x(t_k) + \frac{\partial F}{\partial x} dt + \cdots \]
  which include almost all explicit time evolutions
Convergence: MD (1)

* Convergence property of molecular dynamics simulations
  * time evolution: symplectic integrator
  * a liquid cluster with 249 Ar atoms (~80K)
  * 2nd order symplectic integrator (SI) (Velocity Verlet method):
    \[
    x_j(t + \Delta t) = x_j(t) + \left[ v_j(t) + \frac{F_j(t)}{2m} \Delta t \right] \Delta t
    \]
    \[
    v_j(t + \Delta t) = v_j(t) + \frac{F_j(t) + F_j(t + \Delta t)}{2m} \Delta t
    \]
  * Higher order SIs are constructed by multiple 2nd order SIs.

Convergence: MD (2)

* a is the order of the SI, r is the order of the iParareal.
* Errors by iParareal is limited by that of the original SI.

Convergence: Quantum Mech. (1)

* Quantum time evolutions are represented by unitary transformations:
  \[
  |\psi(t + \Delta t)\rangle = \exp \left( \frac{\Delta t}{i\hbar} H(t) \right) |\psi(t)\rangle \equiv \hat{U}_{\Delta t}(t) |\psi(t)\rangle
  \]
* If you note \(|\psi(t + \Delta t)\rangle = \{ I + [\hat{U}_{\Delta t}(t) - I] \} |\psi(t)\rangle\), it is realized that iParareal is applicable to this kinds of problems.
  * |\psi(t)\rangle, |\psi(t + \Delta t)\rangle : normalized complex vectors
  * \(\hat{U}_{\Delta t}(t)\) : unitary matrix
  * iParareal relation for quantum time-evolutions:
    \[
    |\psi^{(r+1)}(t + \Delta t)\rangle = |\psi^{(r+1)}(t)\rangle + (\hat{U}_{\Delta t}(t) - I) |\psi^{(r)}(t)\rangle
    \]

Convergence: Quantum Mech. (2)

* Because of linearity, its convergence is proven by the use of the spectral radius \(\rho(\hat{U}_{\Delta t}(t) - I)\).
* Errors of \(r\)-th iParareal for matrix-vector multiplications:
  * solid: numerical results, dashed: estimation by \(\rho(\hat{U}_{\Delta t}(t) - I)\)
    \[
    \frac{|x_k - x_k^{(r)}|}{|x_0|} \leq \sum_{j=0}^{r-1} \left( k \atop j \right) \left[ \rho(\hat{U}_{\Delta t}(t) - I) \right]^j
    \]

**Configuration of iParareal**

- Without $g_k(x)$ calculations, we can efficiently configure the time parallel computations.
- The limit of the speedup is extended to $T_f / T_c$

![Diagram of iParareal configuration](image)

**Timelines of iParareal (1)**

- To obtain timelines over multiple nodes:
  - Adjust clocks between different nodes
  - Measure by MPI_Wtime or other time functions
  - Gather and visualize the timeline data
- Visualized example:
  - $|\psi(t)|$, $|\psi(t + \Delta t)|$ 2048 elements
  - $U\Delta t(t)$: 2048 x 2048
  - $P=4, R=4$: $K=P+R-1=7$
  - Speedup: 1.73

**Timelines of iParareal (2)**

- 8 (space) x 16 (time)
  - 4.03 msec x 19 $\Rightarrow$ 16.25 msec
  - Speedup: 4.71
- 16 (space) x 8 (time)
  - 2.19 msec x 11 $\Rightarrow$ 13.20 msec
  - Speedup: 1.82

**Bucket-Brigade Communications**

- The bucket-brigade communication is
  - One dimensional neighbor collective communication with simple calculations
  - Dependent on neighboring processes
  - Implemented by MPI
- $k$-th resource in time direction:
  - Receives $x_k$ from the adjacent resource
  - Calculates $x_{k+1} = x_k + y_k$
  - Transfers $x_{k+1}$ to the next process
**Measurement and Visualization**

- Example: Bucket Brigades over 12 nodes:
  - Procedure of the measurement:
    1. Calibrate clocks
    2. Record by MPI_Wtime
    3. Visualize the result
  - Machine and conditions:
    - CX400: Xeon cluster connected by InfiniBand FDR
    - 1 process in each node

**Performance of Bucket Brigades**

- Results of the measurement:
  - Latency is dominant for small data
  - Large fluctuation is observed

**Non-blocking Communications**

- Blocking Communications by MPI_Recv / Send
  - Sequential execution: MPI_Recv, Calculation, MPI_Send
  - No overlaps between communications and other works
- Non-blocking Communications by MPI_Irecv / Isend
  - Pipelined bucket-brigades by dividing data into smaller one
  - Overlapping with other works

**Speedup by Pipelining**

- Expected speedup:
  \[ S_p = \frac{(T_c + T_a)(K - 1)}{\left(\frac{T_c + T_a}{P} + T_l\right)(K - 1) + \max(T_c, T_a)(P - 1)} \]
  
  - \( T_c \): communication time, \( T_a \): time for calculation (add), \( T_l \): latency of communication, \( K \): # of stages, \( P \): subdivision
Summary & Conclusions

* We introduced `iParareal' to parallelize time evolutions.
  * iParareal made us free from defining the coarse solver $G_k(x)$.
* Convergence and performance of iParareal were shown.
  * Applicable to MD and quantum time evolutions.
  * It is practically used to accelerate time evolutions.
* We analyzed the bucket-brigade communication pattern
  which was abstracted from communications in iParareal.
  * Performance was measured and its timeline was shown.
  * We showed that pipelining is effectively applied for speedup.

Future Works

* To apply the pipelined bucket-brigade communication to
  iParareal method, and to confirm its performance.
* A large-scale computation of the bucket-brigade
  communications and iParareal method.
* To provide iParareal interface and the bucket-brigade
  communication API for application programmers.
* Auto or dynamic tuning functions to obtain appropriate
  parameters for bucket-brigade communications.
  * ...

Thank you for your attention!

* This work is supported by JST CREST.