Assessing the cost of redistribution followed by a computational kernel: complexity and performance results

Julien Herrmann, Georges Bosilca, Thomas Herault, Loris Marchal, Yves Robert and Jack Dongarra

Lyon - Juil. 2, 2014
Outline

1. Introduction
2. Framework
3. Coupling redistribution and computation
4. Performance Evaluation
5. Future work
Data Collection

- Origin of data: sensors (e.g. satellites) that aggregate snapshots
- Stored in a virtual multi-dimensional array:
  - Space (X, Y, [Z])
  - Time
  - Sensor Type
- Raw data are often:
  - sparse: not all tiles are observed at all time
  - skewed: some areas are much better covered than others
Data Storage

- Data elements are stored on different processors
- Computation kernel (e.g. linear algebra kernels) must be applied to data
- Initial data distribution may be inefficient for the computation kernel

Distribution of a tiled matrix
Problem Statement

- A data distribution is usually defined to minimize the completion time of an algorithm.

- There is not necessarily a single data distribution that maximizes this efficiency.

- Find the one-to-one mapping (subsets of data - processors) for which the cost of the redistribution is minimal.
Outline

1. Introduction
2. Framework
3. Coupling redistribution and computation
4. Performance Evaluation
5. Future work
Data distribution / Data partition

- Let $P$ be a finite set of identical processors
- Let $A$ be a finite set of data items

Data Distribution: $\mathcal{D} : A \rightarrow P$

$\forall a \in A, \mathcal{D}(a) = p \iff a \text{ hosted on proc } p$

Data Partition: $\mathcal{P} : A \rightarrow P$

$\forall a, b \in A, \mathcal{P}(a) = \mathcal{P}(b) \iff a \text{ and } b \text{ are hosted by the same processor}$

- A data distribution $\mathcal{D}$ is compatible with the data partition $\mathcal{P}$
  if there exists a permutation $\sigma$ such that
  $\forall a \in A, \mathcal{P}(a) = \sigma(\mathcal{D}(a))$
Cost of redistribution

- **Hardware symmetry assumption**: the efficiency of the computation algorithm is a function of the data partition.

- **Unitary size assumption**: all data items are of the same size.

Evaluation of the redistribution with two metrics:
- **Total volume of communication**: the total number of data items sent from one processor to another.
- **Number of parallel communication steps**: one-port bi-directional model.
Practical approach

- For many algorithms, we know ideal data partitions that minimize completion time.
- There are $P!$ data distributions compatible with the ideal partition.

**Tractable problem**

Given an initial data distribution $D_{ini}$, find the data distribution $D_{tar}$ compatible with the ideal data partition that minimizes the cost of redistribution.
Best redistribution for vol

Algorithm 1: BESTDISTRIBFORVOLUME

**Data:** Initial data distribution $\mathcal{D}_{ini}$ and target data partition $\mathcal{P}_{tar}$

**Result:** a data distribution $\mathcal{D}_{tar}$ compatible with the given data partition so that $\text{RedistVol}(\mathcal{D}_{ini} \rightarrow \mathcal{D}_{tar})$ is minimized

1. $A \leftarrow \{1, \ldots P\}$ (set of processors);
2. $B \leftarrow \{1, \ldots P\}$ (set of data partition components);
3. $G \leftarrow$ complete bipartite graph $(V, E)$ where $V = A \cup B$;
4. foreach edge $(i, j)$ in $E$ do
   - $\text{weight}(i, j) \leftarrow |\{d \in \mathcal{P}_{tar}(j) \text{ s.t. } \mathcal{D}_{ini}(d) \neq i\}|$
5. $\mathcal{M} \leftarrow$ minimum-weight perfect matching of $G$;
6. foreach $(i, j) \in \mathcal{M}$ do
   - for $d \in \mathcal{P}_{tar}(j)$ do $\mathcal{D}_{tar}(d) \leftarrow i$

return $\mathcal{D}_{tar}$

- Create a complete bipartite graph:
  - Left nodes: set of data partition components
  - Right nodes: set of processors
  - Weight of edges: number of communications assigning component $q$ to processor $p$ would create

- Find a minimum-weight perfect matching of this graph
  $\rightarrow O(NP + P^3)$
Best redistribution for step

**Algorithm 2: **BEST_DISTRIFFORSTEPS

**Data:** Initial data distribution $D_{ini}$ and target data partition $P_{tar}$

**Result:** A data distribution $D_{tar}$ compatible with the given data partition so that $\text{RedistSteps}(D_{ini} \rightarrow D_{tar})$ is minimized

- $A \leftarrow \{1, \ldots, P\}$ (set of processors);
- $B \leftarrow \{1, \ldots, P\}$ (set of data partition components);
- $G \leftarrow$ complete bipartite graph $(V, E)$ where $V = A \cup B$;

**for** edge $(i, j)$ in $E$ **do**

- $r_{i,j} \leftarrow |\{d \in P_{tar}(j) \text{ s.t. } D_{ini}(d) \neq i\}|$;
- $s_{i,j} \leftarrow |\{d \in \bigcup_{k \neq j} P_{tar}(k) \text{ s.t. } D_{ini}(d) = i\}|$;
- $\text{weight}(i, j) \leftarrow \max(r_{i,j}, s_{i,j})$

$M \leftarrow$ maximum cardinality matching of $G$ (using the HopcroftKarp Algorithm);

**while** $|M| \neq P$ **do**

- $M_{\text{save}} \leftarrow M$;
- Suppress all edges of $G$ with maximum weight;
- $M \leftarrow$ maximum cardinality matching of $G$ (using the HopcroftKarp Algorithm);

**return** $M_{\text{save}}$

- Create the same complete bipartite graph (under the new redistribution cost assumption)
- Find the maximum cardinality matching with the smallest maximum edge

$\rightarrow O(NP + P^{9/2})$
Communication reduction

- Initial data distribution: random balanced distribution
- Targeted data partition: balanced partition $P_{tar}$
- Reference: arbitrary balanced distribution compatible with $P_{tar}$

$D$ is the number of data items per processor
Outline

1. Introduction
2. Framework
3. Coupling redistribution and computation
4. Performance Evaluation
5. Future work
Redistribution followed by computation kernel

- Non-overlapping phases assumption:
  \[ T_{\text{tot}} = T_{\text{redist}}(D_{\text{ini}} \rightarrow D_{\text{tar}}) + T_{\text{comp}}(D_{\text{tar}}) \]

- Close formula for \( T_{\text{redist}}(D_{\text{ini}} \rightarrow D_{\text{tar}}) \) depending on the communication model

- No close formula for \( T_{\text{comp}}(D_{\text{tar}}) \) in the general case
Consider the simple case of iterative 1D Stencil

**Algorithm 3:** One iteration of the unidimensional stencil algorithm

```plaintext
for 0 ≤ d ≤ N − 1 in parallel do
    ℓ_d ← (d − 1) mod N;
    r_d ← (d + 1) mod N;
    if D(ℓ_d) ≠ D(d) then
        Processor D(d) receives data item ℓ_d from processor D(ℓ_d);
    end if
    if D(r_d) ≠ D(d) then
        Processor D(d) receives data item r_d from processor D(r_d);
    end if
for 0 ≤ d ≤ N − 1 in parallel do
    Processor D(d) updates data item d using ℓ_d and r_d;
```

Simple close formula for $T^{\text{stencil}}_{\text{comp}}(D_{\text{tar}})$ for both communication models
NP-completeness

Theorem
Finding the optimal distribution $D_{tar}$ that minimizes

$$T_{tot} = T_{\text{redist}}(D_{ini} \rightarrow D_{tar}) + T_{\text{stencil}}^{\text{comp}}(D_{tar})$$

is NP-complete in the strong sense.

Proof.
Polynomial reduction from the 3-Partition Problem (i.e. decide whether a given multiset of integers can be partitioned into triples that all have the same sum)
Outline

1. Introduction
2. Framework
3. Coupling redistribution and computation
4. Performance Evaluation
5. Future work
Based on the Parsec environment:

- data flow description of the algorithm
- deals with MPI communications across nodes and with shared memory accesses (threads) inside nodes
- tasks are local

Data-locality independent version of the algorithms:

- the operation takes an initial data distribution and the computation distribution
- Parsec moves data from initial distribution to the computation location while computation happens
Experimental set up

- **Initial distribution**: random balanced distribution
- **Targeted partition**: optimal partition $P_{\text{tar}}$ for the computation kernel
- **Targeted distribution** is computed according to four heuristics:
  - Owner compute (default heuristics of Parsec)
  - Canonical (arbitrary distribution compatible with $P_{\text{tar}}$)
  - Vol ($D_{\text{tar}}$ computed by BestDistribForVolume)
  - Steps ($D_{\text{tar}}$ computed by BestDistribForSteps)
\[ T_{\text{comm}} = T_{\text{calc}} \]

Machine: Dancer, 8x16 cores, IB20G

- Owner compute strategy is not effective for more than one stencil step
- 0 stencil step and owner compute: overhead of the PaRSEC runtime
- 0 stencil step: vol and step heuristics provide 20% improvement over canonical
QR factorization

<table>
<thead>
<tr>
<th>n</th>
<th>Vol. of comm. in the redist. phase</th>
<th>Nb. of steps in the redist. phase</th>
<th>Vol. of comm. in QR fact.</th>
<th>Completion time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>owner</td>
<td>cann</td>
<td>vol</td>
<td>step</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>240</td>
<td>205</td>
<td>233</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>1,087</td>
<td>1,004</td>
<td>1,072</td>
</tr>
<tr>
<td>52</td>
<td>0</td>
<td>2,526</td>
<td>2,425</td>
<td>2,518</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>4,598</td>
<td>4,459</td>
<td>4,575</td>
</tr>
<tr>
<td>88</td>
<td>0</td>
<td>7,271</td>
<td>7,129</td>
<td>7,242</td>
</tr>
</tbody>
</table>

- Average results on 50 matrices
- The three redistributing strategies moved around 90% of the tiles
- Owner-compute requires less fewer communication during the QR factorization
- But the three other strategies lead to a 10-15% improvement on the total completion time
Outline

1. Introduction
2. Framework
3. Coupling redistribution and computation
4. Performance Evaluation
5. Future work
Conclusion and future work

Conclusion:
- Algorithms that find the optimal target distribution for different redistribution metrics
- NP-completeness proof for minimizing redistribution time followed by a computation kernel
- Experimental validation for Stencil and QR factorization kernels

Future work:
- Target not only the optimal partition
- Special case of the Earth Science applications