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

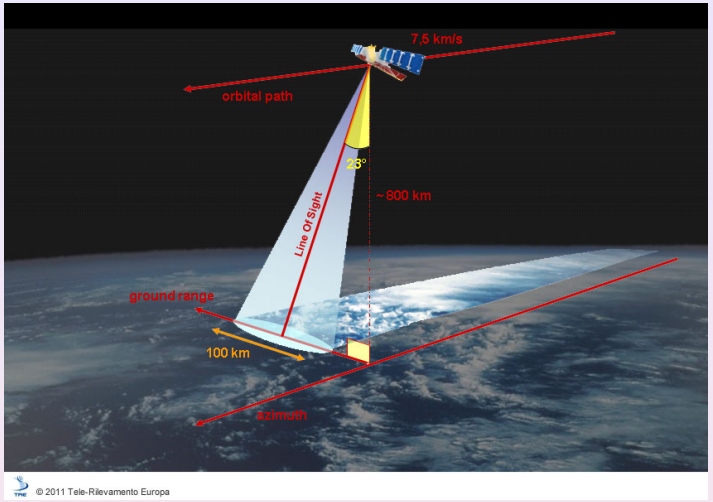
Lyon - Juil. 2, 2014

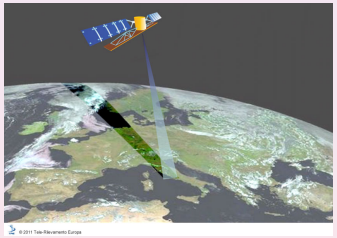
Outline

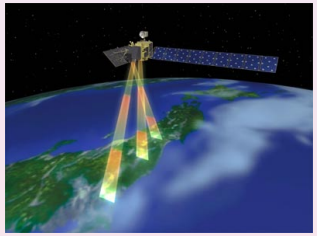
Outline

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

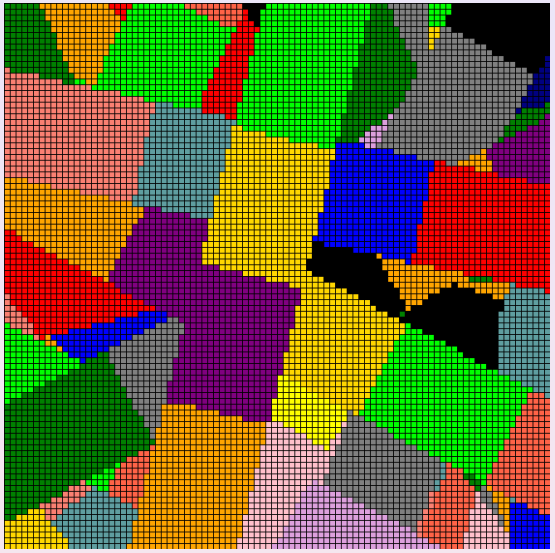






Data Storage

- Data elements are stored on different processors
- Computation kernel (e.g. linear algebra kernels) must be applied to data

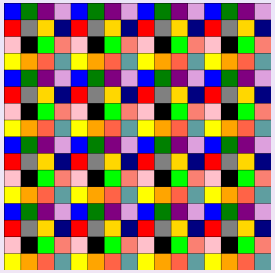


Distribution of a tiled matrice

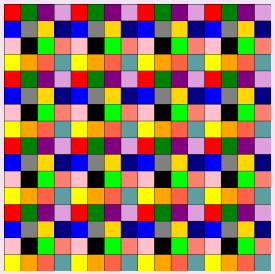
- Initial data distribution may be inefficient for the computation kernel

Problem Statement

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



- There is not necessary 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

Data distribution / Data partition

- Let P be a finite set of identical processors
- Let A be a finite set of data items
- Data Distribution: $\mathcal{D} : A \mapsto P$
  $\forall a \in A, \mathcal{D}(a) = p \Leftrightarrow$ a hosted on proc p
- Data Partition: $\mathcal{P} : A \mapsto P$
  $\forall a, b \in A, \mathcal{P}(a) = \mathcal{P}(b) \Leftrightarrow$ a and b are hosted by the same processor
- A data distribution $\mathcal{D}$ is compatible with the data partition $\mathcal{P}$ iif there exists a permutation σ 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 $\mathcal{D}_{ini}$, find the data distribution $\mathcal{D}_{tar}$ 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 $RedistVol(\mathcal{D}_{ini} \rightarrow \mathcal{D}_{tar})$ 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 $V = A \cup B$;
foreach *edge* (i, j) *in* E **do**
  $weight(i, j) \leftarrow |\{d \in \mathcal{P}_{tar}(j) \text{ s.t. } \mathcal{D}_{ini}(d) \neq i\}|$
$\mathcal{M} \leftarrow$ minimum-weight perfect matching of G;
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 $\hookrightarrow O(NP + P^3)$

Best redistribution for step

Algorithm 2: BESTDISTRIBFORSTEPS

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 $RedistSteps(\mathcal{D}_{ini} \rightarrow \mathcal{D}_{tar})$ 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**
  $r_{i,j} \leftarrow |\{d \in \mathcal{P}_{tar}(j) \text{ s.t. } \mathcal{D}_{ini}(d) \neq i\}|$;
  $s_{i,j} \leftarrow |\{d \in \bigcup_{k \neq j} \mathcal{P}_{tar}(k) \text{ s.t. } \mathcal{D}_{ini}(d) = i\}|$;
  $weight(i, j) \leftarrow \max(r_{i,j}, s_{i,j})$

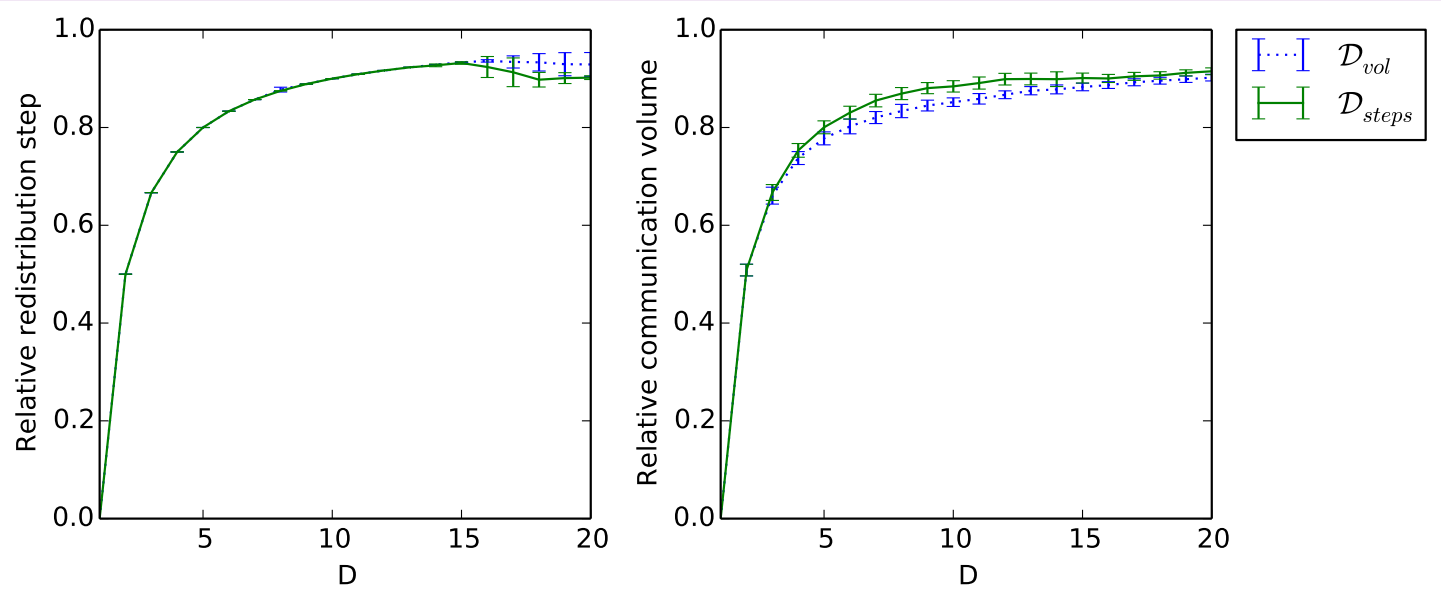
$\mathcal{M} \leftarrow$ maximum cardinality matching of G (using the HopcroftKarp Algorithm);
while $|\mathcal{M}| == P$ **do**
  $\mathcal{M}_{save} \leftarrow \mathcal{M}$;
  Suppress all edges of G with maximum weight;
  $\mathcal{M} \leftarrow$ maximum cardinality matching of G (using the HopcroftKarp Algorithm);

return $\mathcal{M}_{save}$

- Create the same complete bipartite graph (under the new redistribution cost assumption)
- Find the maximum cardinality matching with the smallest maximum edge
  $\hookrightarrow O(NP + P^{\frac{9}{2}})$

Communication reduction

- Initial data distribution: random balanced distribution
- Targeted data partition: balanced partition $\mathcal{P}_{tar}$
- Reference: arbitrary balanced distribution compatible with $\mathcal{P}_{tar}$



- D is the number of data items per processor

Outline

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

1D Stencil

- Consider the simple case of iterative 1D Stencil

Algorithm 3: One iteration of the unidimensional stencil algorithm

for $0 \leq d \leq N-1$ *in parallel* **do**
  $\ell_d \leftarrow (d-1) \mod N$;
  $r_d \leftarrow (d+1) \mod N$;
  if $\mathcal{D}(\ell_d) \neq \mathcal{D}(d)$ **then**
    Processor $\mathcal{D}(d)$ receives data item ℓ_d from processor $\mathcal{D}(\ell_d)$;
  if $\mathcal{D}(r_d) \neq \mathcal{D}(d)$ **then**
    Processor $\mathcal{D}(d)$ receives data item r_d from processor $\mathcal{D}(r_d)$;

for $0 \leq d \leq N-1$ *in parallel* **do**
  Processor $\mathcal{D}(d)$ updates data item d using ℓ_d and r_d;

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

NP-completeness

Theorem

Finding the optimal distribution $\mathcal{D}_{tar}$ that minimizes

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

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

Parsec

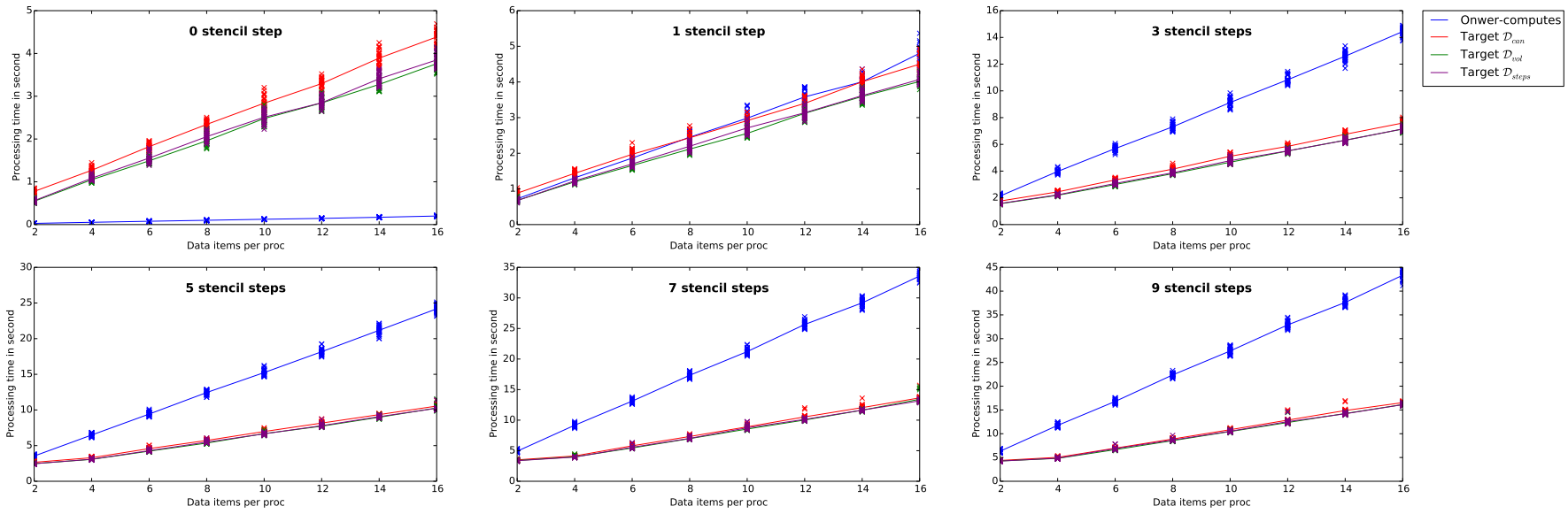
- Based on the Parsec environment:
  $\hookrightarrow$ data flow description of the algorithm
  $\hookrightarrow$ deals with MPI communications across nodes and with shared memory accesses (threads) inside nodes
  $\hookrightarrow$ tasks are local

- Data-locality independent version of the algorithms:
  $\hookrightarrow$ the operation takes an initial data distribution and the computation distribution
  $\hookrightarrow$ 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 $\mathcal{P}_{tar}$ for the computation kernel
- Targeted distribution is computed according to four heuristics:
  - Owner compute (default heuristics of Parsec)
  - Canonical (arbitrary distribution compatible with $\mathcal{P}_{tar}$)
  - Vol ($\mathcal{D}_{tar}$ computed by BestDistribForVolume)
  - Steps ($\mathcal{D}_{tar}$ computed by BestDistribForSteps)

$Tcomm = Tcalc$



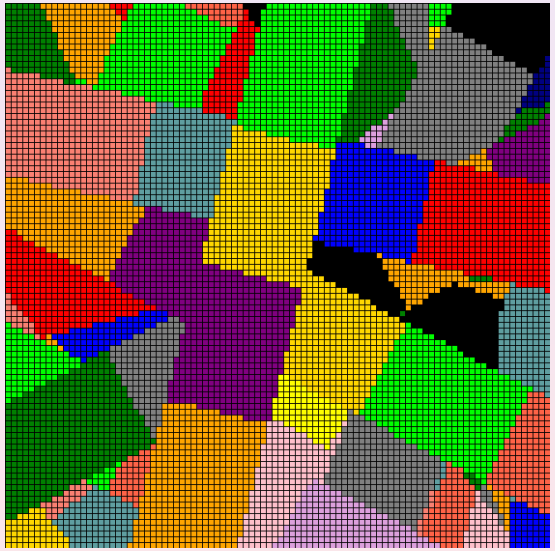
Machine : Dancer, 8x16 cores, IB20G

- 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

	Vol. of comm. in the redist. phase				Nb. of steps in the redist. phase				Vol. of comm. in QR fact.		Completion time			
n	owner	cann	vol	step	owner	cann	vol	step	owner	cann-vol-step	owner	cann	vol	step
16	0	240	205	233	0	34	31	31	1140	831	2.63	1.92	1.89	1.89
34	0	1, 087	1, 004	1, 072	0	153	142	142	6, 380	5, 720	11.05	8.77	8.23	8.61
52	0	2, 526	2, 425	2, 518	0	349	338	338	19, 295	16, 669	30.22	26.12	22.41	22.31
70	0	4, 598	4, 459	4, 575	0	681	660	659	29, 606	37, 942	61.16	53.41	52.29	58.21
88	0	7, 271	7, 129	7, 242	0	963	951	951	45, 311	69, 951	114.69	100.88	96.70	99.36

- 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



Distribution of a tiled matrice

Outline

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