Performance Models for Data Transfers: A Case Study with Molecular Chemistry Kernels

Suraj KUMAR[1], Lionel EYRAUD-DUBOIS[2], and Sriram KRISHNAMOORTHY[1]

[1]Pacific Northwest National Laboratory, Richland, USA

[2]Inria Bordeaux – Sud-Ouest, France

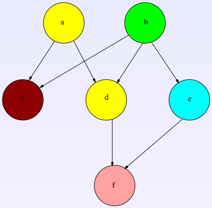
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

August 8, 2019

Introduction

- Distributed memory systems are very common
- Rate of computation vs rate of data movement
- Focus: Avoid, Hide, and Minimize communication costs

Task Graphs and Runtime Systems

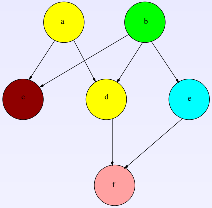


- Applications can be expressed as task graphs
- Vertices represent tasks
- Edges represent dependencies among tasks

Task Based Runtime Systems

- StarPU, QUARK, PaRSEC, StarSs, Legion, KAAPI
- May only see a set of ready (independent) tasks
- A memory node may require data from other memory nodes
- Order of data transfers such that communication-computation overlap is maximized

Task Graphs and Runtime Systems



- Applications can be expressed as task graphs
- Vertices represent tasks
- Edges represent dependencies among tasks

Task Based Runtime Systems

- StarPU, QUARK, PaRSEC, StarSs, Legion, KAAPI
- May only see a set of ready (independent) tasks
- A memory node may require data from other memory nodes
- Order of data transfers between two memory nodes such that communication-computation overlap is maximized

Problem Definition

- Communication and computation times are known in advance
  - Approximated based on number of computations and hardware details
  - Obtained from previous executions

Problem *DT*

- A set of tasks $ST = \{T_1, \cdots, T_n\}$ is scheduled on a processing unit P with memory unit M of capacity C
- Input data for tasks of ST reside on another memory unit
- Tasks do not produce output data
- Tasks do not require intermediate memory
- A tasks uses an amount of memory in M from the start of its communication to the end of its computation

Given L, is there a feasible schedule S for ST such that makespan of S, $\mu(S) \leq L$?

Relevant Problem in Literature

Machine Flowshop Problem

- n machines and m tasks
- Each task contains exactly n operations
- i-th operation of a task must be processed on the i-th machine
- Each machine can perform at-max one operation at a time
- i-th operation of a task starts after the completion of $(i-1)$-th operation

The problem is to obtain the arrangement that achieves shortest possible makespan.

- Johnson provided an optimal algorithm for 2-machine flow shop problem
- Our problem DT adds one extra dimension (memory capacity) to 2-machine flowshop problem

Table of Contents

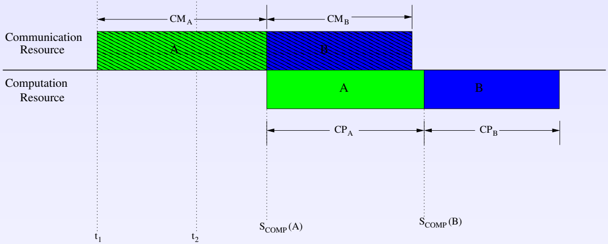
Unlimited Memory Capacity

Johnson's Algorithm

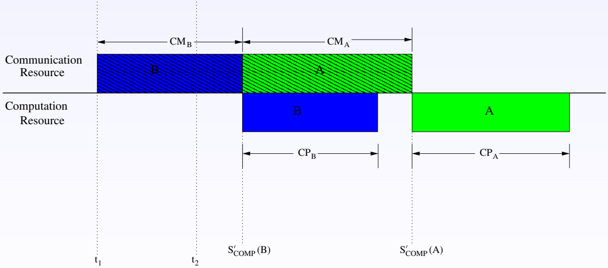
```
Divide ready tasks in two sets S_1 and S_2. If computation time of
  a task T is not less than its communication time, then T is in S_1
  otherwise in S_2.
Sort S_1 in queue Q by non-decreasing communication times
Sort S_2 in queue Q' by non-increasing computation times
Append Q' to Q
τ_COMM ← 0                {Available time of communication resource}
τ_COMP ← 0                {Available time of computation resource}
while Q ≠ ∅ do
    Remove a task T from beginning of Q for processing
    S_COMM(T) ← τ_COMM
    S_COMP(T) ← max(S_COMM(T) + COMM_T, τ_COMP)
    τ_COMM ← S_COMM(T) + COMM_T
    τ_COMP ← S_COMP(T) + COMP_T
end while
```

- $OMIM$ denotes the makespan of this algorithm

Approach for the optimality proof



Original Schedule



Swapped Schedule

Table of Contents

Limited Memory Capacity

- Memory is required only to store input data (from the definition of our problem)
- As Communication time $\propto$ amount of communication, for each task:
  - Communication time = Amount of communication (for simplification)
  - Communication time = Amount of input data

Reduction Problem

Three Partition Problem (3PAR): Given a set of $3m$ integers $A = \{a_1, \cdots, a_{3m}\}$, is there a partition of A into m triplets $TR_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$, such that $\forall i, a_{i_1} + a_{i_2} + a_{i_3} = b$, where $b = (1/m) \sum a_i$?

Reduction: 3PAR to DT

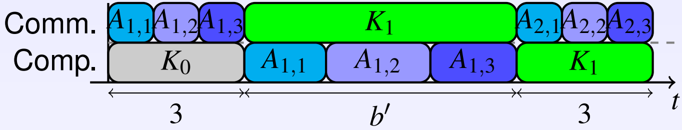
Definition of tasks in the reduction from 3PAR

$x = max\{a_i : 1 \leq i \leq 3m\}$

Task	Communication time	Computation time
K_0	0	3
$K_1, \cdots, K_{m-1}$	$b' = b + 6x$	3
K_m	$b' = b + 6x$	0
$1 \leq i \leq 3m, A_i$	1	$a'_i = a_i + 2x$

Total memory capacity: $C = b' + 3$
Target makespan: $L = m(b' + 3)$

Pattern of Feasible Schedule

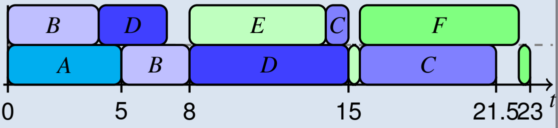


- Problem DT is NP-Complete.
- Our proof is inspired from work by Papadimitriou et al. (on 2-machine flowshop with limited buffer problem)

Order of Processing on Communication and Computation Resources in Optimal Schedules

Task	Memory Req	Comm Time	Comp Time
A	0	0	5
B	4	4	3
C	1	1	6
D	3	3	7
E	6	6	0.5
F	7	7	0.5

Common ordering on both resources



Different ordering on both resources

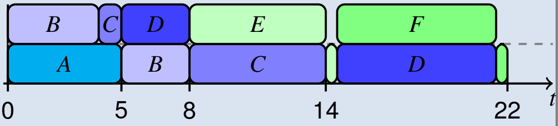


Table of Contents

Our Heuristics

- Static order based strategies
- Dynamic selection based strategies
- Static order with dynamic correction based strategies

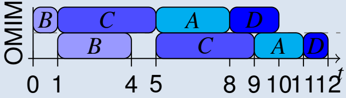
We consider common order on both resources for all of our heuristics.

Static Order Based Strategies

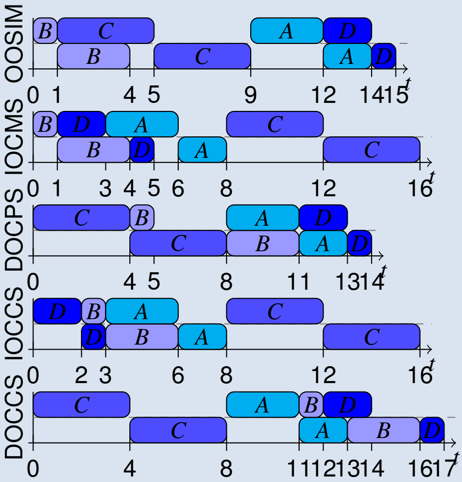
- *order of optimal strategy infinite memory* ($OOSIM$)
- *increasing order of communication strategy* ($IOCMS$)
- *decreasing order of computation strategy* ($DOCPS$)
- *increasing order of communication plus computation strategy* ($IOCCS$)
- *decreasing order of communication plus computation strategy* ($DOCCS$)

Task	Memory Req	Comm Time	Comp Time
A	3	3	2
B	1	1	3
C	4	4	4
D	2	2	1

Unlimited Memory Capacity



Memory Capacity: 6

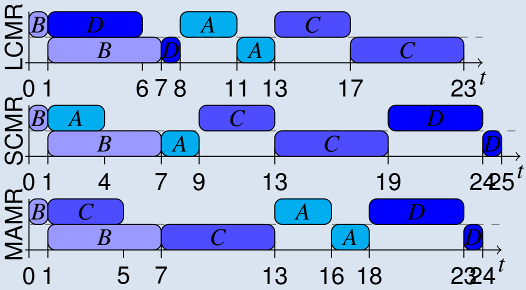


Dynamic Selection Based Strategies

- *largest communication task respects memory restriction* ($LCMR$)
- *smallest communication task respects memory restriction* ($SCMR$)
- *maximum accelerated task respects memory restriction* ($MAMR$)

Task	Memory Req	Comm Time	Comp Time
A	3	3	2
B	1	1	6
C	4	4	6
D	5	5	1

Memory Capacity = 6



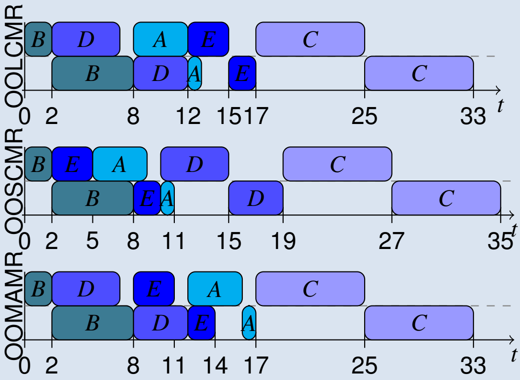
Static Order with Dynamic Corrections Based Strategies

- *optimal order infinite memory largest communication task respects memory restriction* ($OOLCMR$)
- *optimal order infinite memory smallest communication task respects memory restriction* ($OOSCMR$)
- *optimal order infinite memory maximum accelerated task respects memory restriction* ($OOMAMR$)

Static Order with Dynamic Corrections Based Strategies

Task	Memory Req	Comm Time	Comp Time
A	4	4	1
B	2	2	6
C	8	8	8
D	5	5	4
E	3	3	2

Memory Capacity=9



Favorable Situations for Heuristics

Heuristic	Favorable Situation
OOSIM	Memory capacity is not a restriction (Optimal)
IOCMS	Memory capacity is not a restriction and tasks are compute intensive (Optimal)
DOCPS	Memory capacity is not a restriction and tasks are communication intensive (Optimal)
IOCCS	Moderate memory capacity and most tasks are highly compute intensive
DOCCS	Moderate memory capacity and most tasks are highly communication intensive
LCMR	Limited memory capacity and significant percentage of tasks with large communication times are compute intensive
SCMR	Limited memory capacity and significant percentage of tasks with small communication times are compute intensive
MAMR	Limited memory capacity and significant percentage of all types of tasks
OOLCMR	Moderate memory capacity and significant percentage of slightly communication intensive tasks have large communication times
OOSCMR	Moderate memory capacity and significant percentage of slightly communication intensive tasks have small communication times
OOMAMR	Moderate memory capacity and significant percentage of all types of tasks

Additional Heuristics from Previous Work

Gilmore-Gomory (GG)

- A classical algorithm for 2-machine no-wait flowshop problem
- Problem is represented by a graph
- An optimal sequence of vertices is obtained
- Does not take memmory constraints into account

Bin Packing (BP)

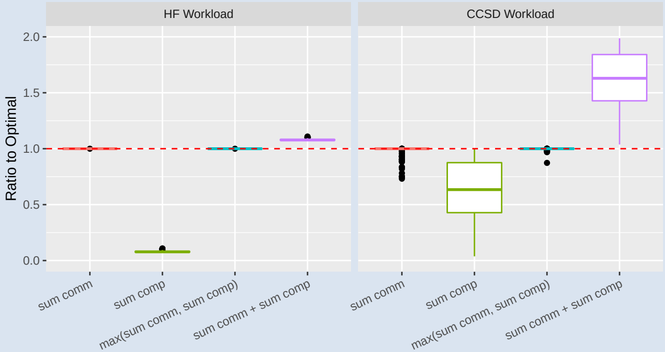
- First-Fit algorithm for the bin packing problem
- Tasks added to the first bin in which they fit
- If no bin is found then a new bin is created
- Sequence made of all tasks from consecutive bins

Table of Contents

Workload Characteristics

Molecular Chemistry Kernels

- Hartree–Fock (HF) with SiOSi molecules and Coupled Cluster Singles Doubles (CCSD) with Uracil molecules
- Tensor operations: transpose and contraction



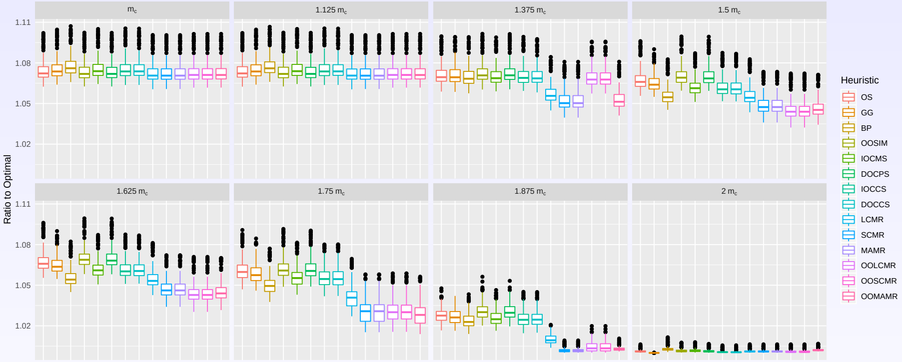
Configuration Parameters to Obtain Traces

- 10 nodes of Cascade machine
- Each node contains 16 Intel Xeon E5-2670 cores
- Double precision version of HF and CCSD of NWChem
- NWChem takes advantages of Global Arrays (GA)
- 150 processes for each application, 300-800 tasks for each process

Simulation Parameters

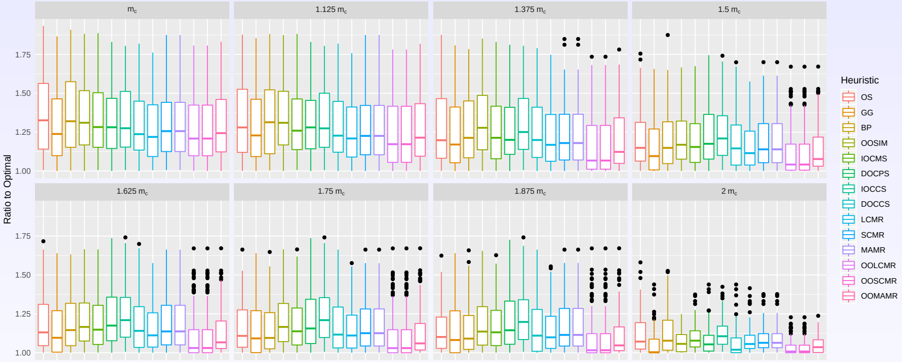
- m_c: minimum memory capacity requirement to execute all tasks of an application
- Evaluation criteria for a heuristic H, $r(H) = \frac{\text{makespan of } H}{OMIM}$ (lower values are better)
- All data transfers between the local memory of each process and the GA memory take the same route

HF Performance with $m_c = 176KB$.



- Dynamic strategies are best suited for limited memory capacities
- Static order with dynamic correction variants outperform others for moderate memory capacities

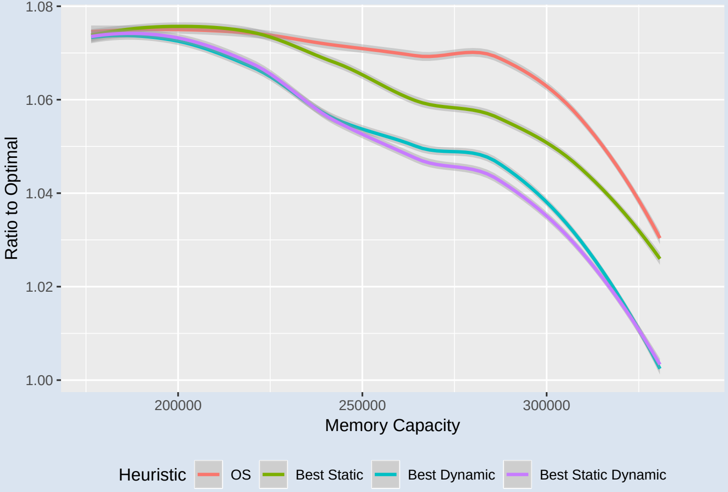
CCSD performance with $m_c = 1.8GB$



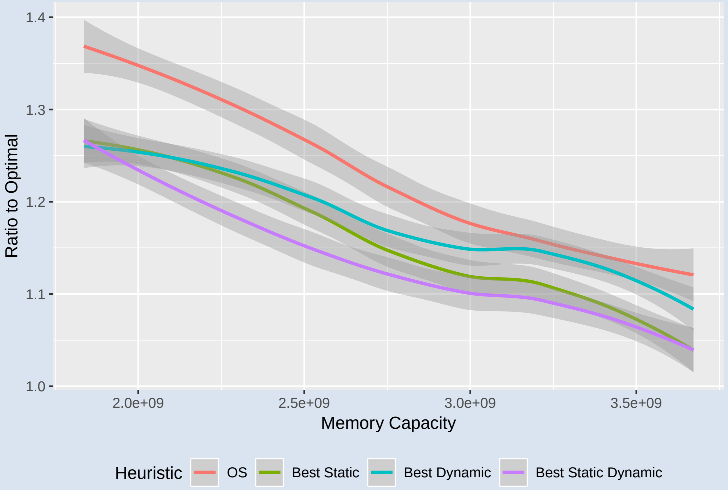
- Highly heterogeneous tasks
- Static order with dynamic correction variants outperform others in most cases

Best variants of all categories

HF Performance



CCSD Performance



Implementation challenges

- Impact of Congestion on communication/computation times
- Output data of a task
- Routes between two memory nodes
- Communications from multiple memory nodes can happen at the same time

Table of Contents

Conclusion and Ongoing Work

Conclusion:

- Problem of determining the optimal order of data transfers is NP-complete
- Our heuristics achieve significant overlap for HF and CCSD applications

Ongoing work:

- Evaluation of our heuristics in the context of accelerators
- Implementation of the proposed heuristics
- Automatic selection of the best heuristic
- Model bandwidth sharing to support multiple simultaneous communications