Latency-Aware Task Assignment and Scheduling in Collaborative Cloud Robotic Systems

An Integer Linear Programming Approach

Shenghui Li[1], Zhiheng Zheng[1], Wuhui Chen[1], Zibin Zheng[1], Junbo Wang[2]

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China

[2]Graduate School of Computer Science and Engineering Aizu-Wakamatsu, Japan

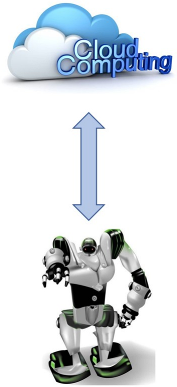
Outline

1. Background of Cloud Robotics
2. Mapping Problem and Algorithm
3. Experimental Results
4. Future Work

Background of Cloud Robotics

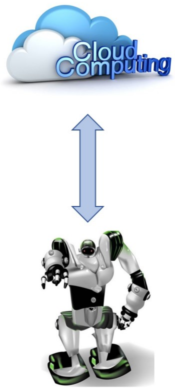
Background

Take advantage of the Internet as a resource for massively parallel computation and resources.



Background

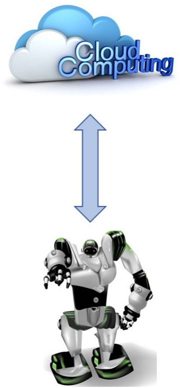
Take advantage of the Internet as a resource for massively parallel computation and resources.



- Cloud provides a shared knowledge database
- Robots can download new skills instantly
- Or offload compute intensive task to the cloud
  - Image processing
  - Real-time Tracking

Background

Take advantage of the Internet as a resource for massively parallel computation and resources.



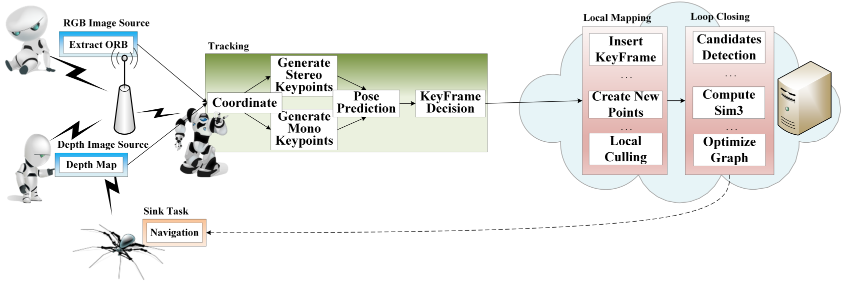
- Cloud provides a shared knowledge database
- Robots can download new skills instantly
- Or offload compute intensive task to the cloud
  - Image processing
  - Real-time Tracking

Advantages

- Cheaper, lighter hardware
- Longer battery life
- Less need for software pushes/updates

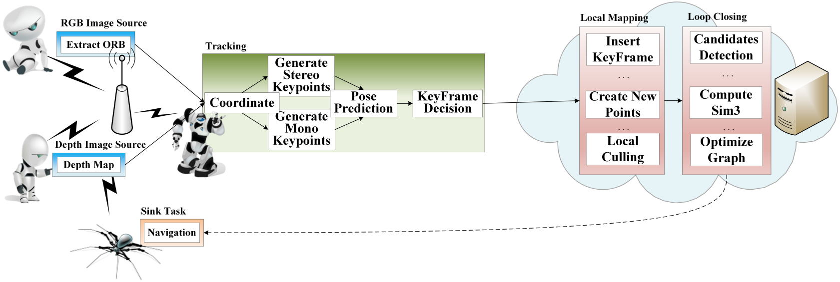
...

Simultaneous Localization And Mapping (SLAM)



Offloading in Cloud Robotics

Simultaneous Localization And Mapping (SLAM)



Challenges

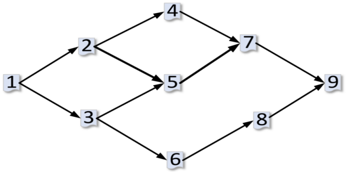
- Overhead of big data delivery, i.e., continuously collecting and processing sensing data
- Transfer rate of the cloud robotic network
- The existence of heterogeneity of robots

Mapping Problem and Algorithm

Task Graph Model

Use a Directed Acyclic Graph (DAG) to represent the application model:

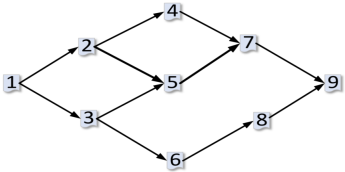
- **Vertices:** task steps
- **Edges:** data dependency
- **Weight of Vertices:** computational complexity
- **Weight of Edges:** intermediate data size



Task Graph Model

Use a Directed Acyclic Graph (DAG) to represent the application model:

- **Vertices:** task steps
- **Edges:** data dependency
- **Weight of Vertices:** computational complexity
- **Weight of Edges:** intermediate data size



Problem

Map the task graph to the compute nodes, minimize the overall makespan.

– *assignment*

– *scheduling*

Formulation

TABLE I: Notations

Constants	
S_{ij}	The size of intermediate data from task i to task$j$
R_{ik}	The running time of task i on node k
B_{kl}	The transmission rate from node k to node l
$\mathcal{T}$	The set of time slots $\mathcal{T} = \{1, 2, ..., t\}$
$pred_i$	The predecessors of task i
$succ_i$	The sucessors of task i
Variables	
x_{ik}	A binary variable indicating whether task i is assigned to node k
x^l_{ik}	A binary variable indicating whether task i is scheduled to run on node k starting at time slot l
y_{ij}	A binary variable indicating whether task i is scheduled to run after j
t^s_i	The start time of task i
t^e_i	The finish time of task i

$$min: \quad t^e_{sink}$$

$$s.t.: \sum_{k \in V_n} x_{ik} = 1, \forall i \in V_t. \tag{1}$$

$$y_{ij} + y_{ji} = 1, \forall i, j \in V_t. \tag{2}$$

$$x_{ik} \leq Z_{ik}, \forall i \in V_t, k \in V_n. \tag{3}$$

$$t^s_i \geq \max_{j \in pred_i} \{ t^e_j + \sum_{(k,l) \in E_n} \frac{x_{ik} x_{jl} S_{ji}}{B_{kl}} \}, \forall i \in V_t, \tag{4}$$

$$t^s_j \geq x_{ik} x_{jk} y_{ij} t^e_i, \forall i \in V_t, \forall j \in V_t, \forall k \in V_n \tag{5}$$

$$t^e_i = t^s_i + \sum_{k \in V_n} x_{ik} R_{ik}, \forall i \in V_t, \tag{6}$$

- Const. (1) ensures that each task is assigned to one compute node
- Const. (2) specifies the global scheduling sequence
- Const. (4) specifies the task and data dependency
- Const. (5) ensures the non-overlapped assumption
- Const. (6) expresses the relationship of start time and finish time

Reformulate to ILP (1/2)

Nonlinear Constraint

$$t_i^s \geq \max_{j \in pred_i} \{t_j^e + \sum_{(k,l) \in E_n} \frac{x_{ik} x_{jl} S_{ji}}{B_{kl}}\}, \forall i \in V_t, \qquad (4)$$

Reformulate to ILP (1/2)

Nonlinear Constraint

$$t_i^s \geq \max_{j \in pred_i} \{t_j^e + \sum_{(k,l) \in E_n} \frac{x_{ik} x_{jl} S_{ji}}{B_{kl}}\}, \forall i \in V_t, \tag{4}$$

- Since x_{ik} is a binary variable, define a new variable f_{ijkl} as:

$$f_{ijkl} = x_{ik} x_{jl}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n, \tag{7}$$

- which can be equivalently replaced by:

$$0 \leq f_{ijkl} \leq x_{ik}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n, \tag{8}$$

$$x_{ik} + x_{jl} - 1 \leq f_{ijkl} \leq x_{jl}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n. \tag{9}$$

Reformulate to ILP (1/2)

Nonlinear Constraint

$$t_i^s \geq \max_{j \in pred_i} \{t_j^e + \sum_{(k,l) \in E_n} \frac{x_{ik} x_{jl} S_{ji}}{B_{kl}}\}, \forall i \in V_t, \qquad (4)$$

- Since x_{ik} is a binary variable, define a new variable f_{ijkl} as:

$$f_{ijkl} = x_{ik} x_{jl}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n, \qquad (7)$$

- which can be equivalently replaced by:

$$0 \leq f_{ijkl} \leq x_{ik}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n, \qquad (8)$$

$$x_{ik} + x_{jl} - 1 \leq f_{ijkl} \leq x_{jl}, \quad \forall (i,j) \in E_t, \forall (k,l) \in E_n. \qquad (9)$$

Conversion

Constraint (3) can be rewritten in a linear form as:

$$t_i^s \geq \max_{j \in pred_i} \{t_j^e + \sum_{(k,l) \in E_n} \frac{f_{ijkl} S_{ji}}{B_{kl}}\}, \forall i \in V_t,$$

Reformulate to ILP (2/2)

Nonlinear Constraint

$$t_j^s \geq x_{ik} x_{jk} y_{ij} t_i^e, \forall i \in V_t, \forall j \in V_t, \forall k \in V_n \tag{5}$$

Reformulate to ILP (2/2)

Nonlinear Constraint

$$t_j^s \geq x_{ik} x_{jk} y_{ij} t_i^e, \forall i \in V_t, \forall j \in V_t, \forall k \in V_n \tag{5}$$

- Discretize continuous time structure to a sequence of time slots
- Define another variable x_{jk}^l to indicate whether task i is scheduled run on node k starting at slot l
- Replace Constraint (5) by:

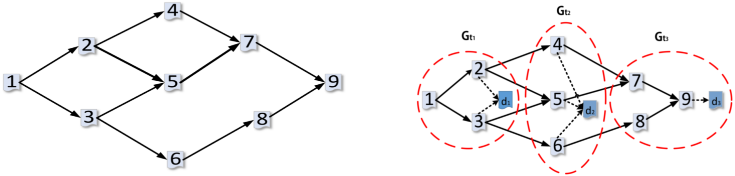
$$\sum_{k \in V_n} \sum_{l \in \mathcal{D}} x_{ik}^l = 1, \forall i \in V_t, \tag{10}$$

$$\sum_{l \in \mathcal{D}} x_{ik}^l = x_{ik}, \forall i \in V_t. \tag{11}$$

$$t_i^s = \sum_{k \in V_n} \sum_{l \in \mathcal{D}} x_{ik}^l T_l, \forall i \in V_t. \tag{12}$$

$$\sum_{i \in V_t} \sum_{v=max(1,l-R_{ik}+1)}^{l} x_{ik}^v \quad \leq 1, \forall i \in V_t, \tag{13}$$

Scaling Approach



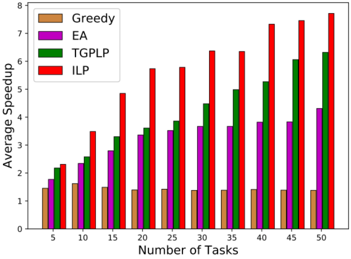
(a) (b)

Task Graph Partitioning

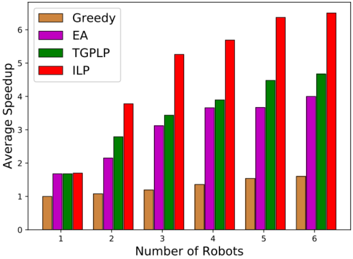
- Partition task graph to smaller subgraphs
- Solve subproblems in which some parts of the problem have been fixed

Experimental Results

Experimental Results



(a) Num. of tasks



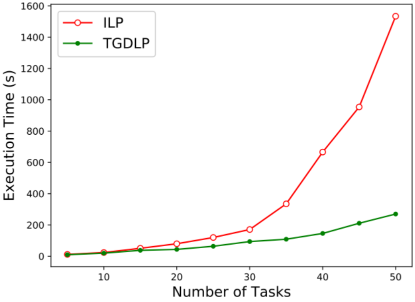
(b) Num. of robots

Performance Comparison

Observation

- ILP approach finds optimal solutions but TGPLP doesn't
- As the number of tasks/robots increases, TGPLP shows its advantage

Experimental Results



Scalability Comparison

Observation

- It is quite time-consuming to get an optimal solution using ILP
- TGPLP approach, provides 'good-enough' results but within acceptable time

Future Work

Future Work

- The method can be further improved by the introduction of heuristic based hot-start solutions;
- We intend to take into consideration some real-world applications under cloud robotic systems;
- And also develop online algorithms to adapt the scenarios with high dynamicity of moving robots.