On the Design of Fault Tolerant Scheduling Algorithm to Reduce the Resource Usage

Laiping Zhao, Kouichi Sakurai
Information Technology and Security Laboratory
Kyushu University

The first author of this research is supported by the governmental scholarship from China Scholarship Council.

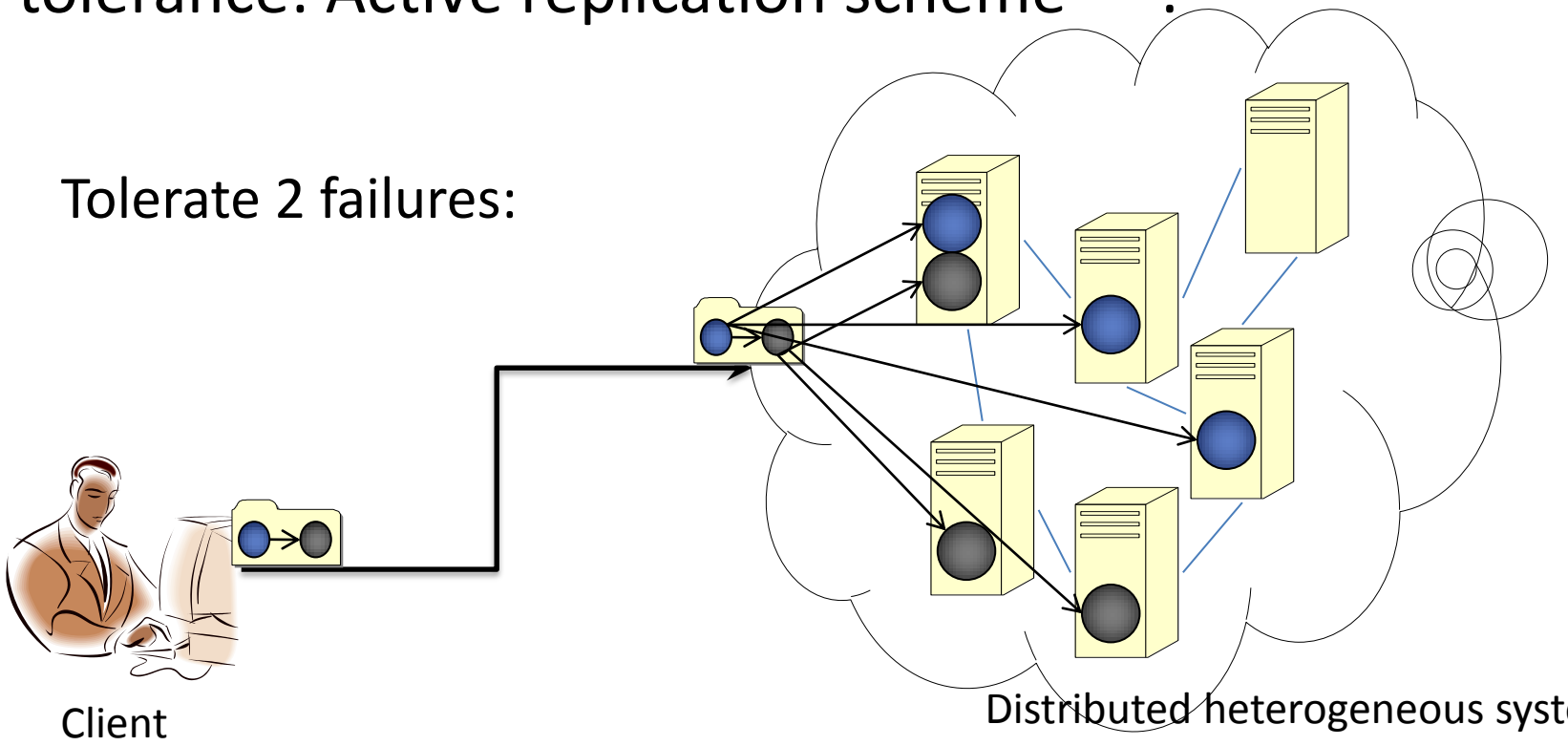
Outline

- 1. Previous work
  - Active replication scheme
  - MaxRe algorithm
  - Problems?
- 2. Before scheduling?
  - Subdeadline assignment
  - Reliability assignment
  - Task priority
- 3. Our proposal: DRR algorithm
- 4. Analysis & Experiments
- 5. Conclusion and future works

1. Previous Work

Fault tolerance: Active replication scheme[1-2]:

Tolerate 2 failures:



The job will be completed even with maximum 2 failures occur.

[1] A. Benoit, et al. Parallel Computing, 2009.
[2] A. Girault et al. DSN2003.

Active Replication Scheme

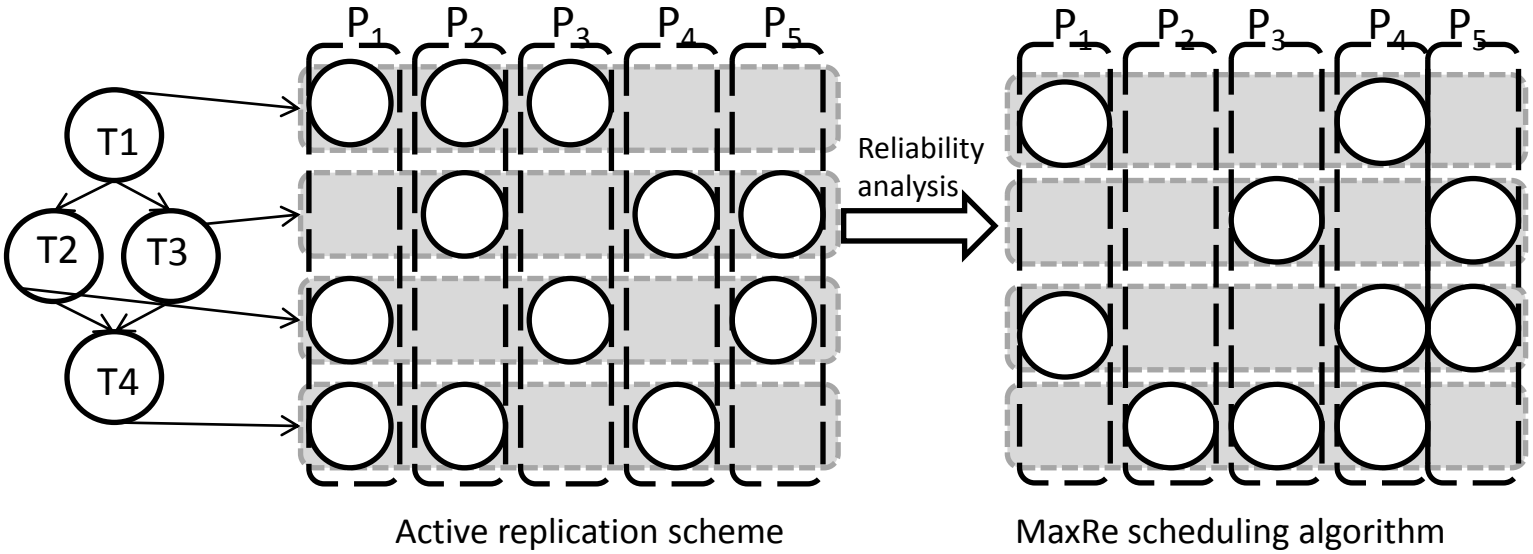
- In order to tolerate *t* failures, the active replica scheme has to schedule *t+1* processors for each task.
- The large resource redundancy problem:
  - Energy consumption
    - Electric power, etc.
    - According to EPA (Environmental Protection Agency) report 2007, Servers and data centers consumed 1.5 percent of total U.S. electricity consumption in 2006.
  - Economic cost
    - To store x gigabytes data, it has to use 3x gigabytes storage space. Assuming the economic cost of each drive is y, the extra 2y cost is believed to be passed on to the customers eventually.
    - E.g. Facebook utilizes 4 replicas, and Google utilizes 3 replicas for each data volume.
- The problem is how to achieve the corresponding reliability with less resources (replicas).

MaxRe Algorithm

The basic idea of the **MaxRe** scheduling algorithm:

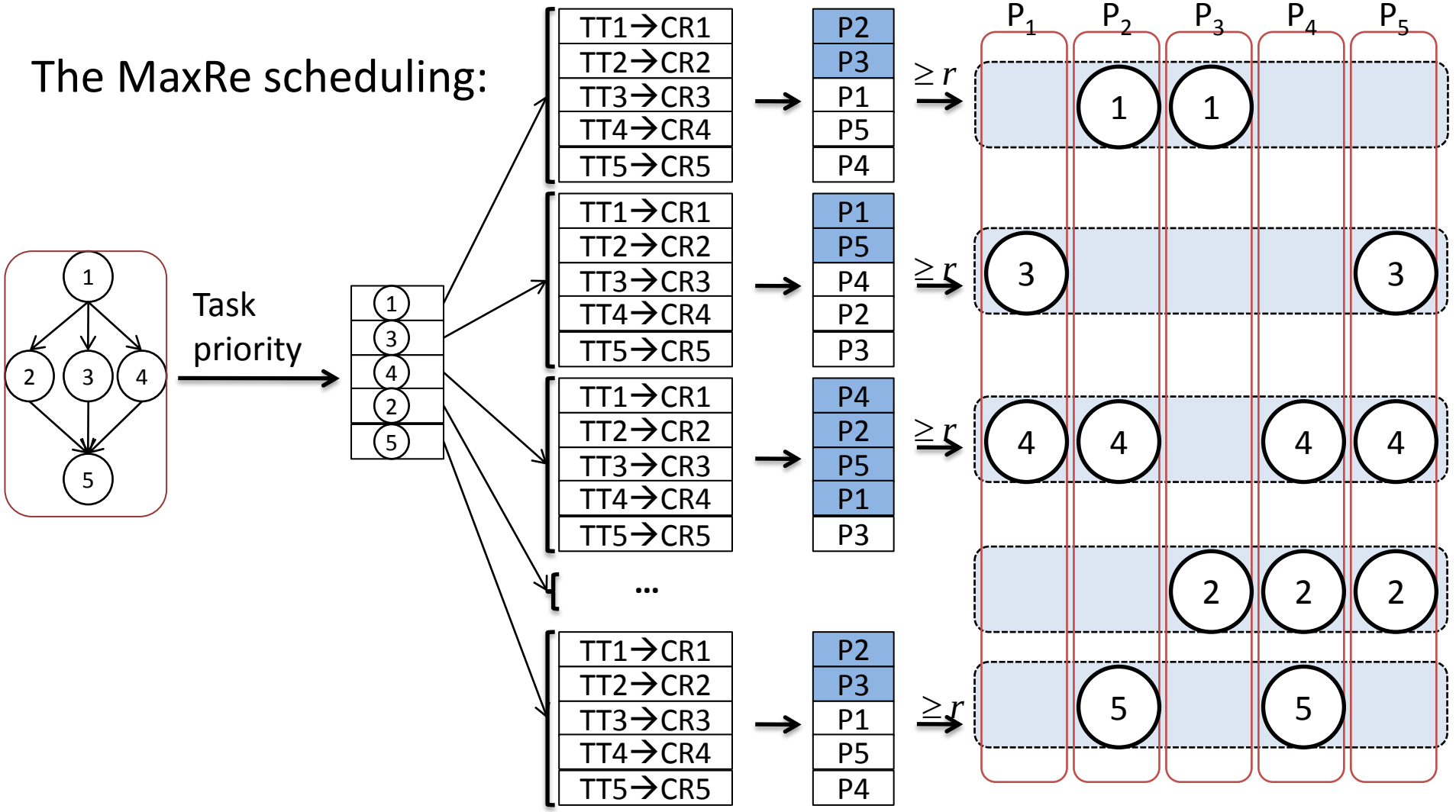
Combining the processors' reliability analysis within the replica scheduling, and make use of less resources to achieve the 2-failures tolerating comparable reliability.

Tolerate 2 failures:



MaxRe Algorithm

The MaxRe scheduling:



MaxRe Algorithm

- Analysis:

Assume that for every task, the deserved number of replicas does not exceed n, and denote the reliability provided by MaxRe as $\mathcal{R}$, then $\mathcal{R} \geq R$.

Proof (key point):

$$R = r^n <= \prod_{i=0}^{n-1} 1 - \prod_{k=0}^{k<\xi} (1 - CR(\tau_i, p_k))$$

$CR(\tau_i, p_k)$ implies the probability that, for task i, the replica p_k does not encounter failures from the very beginning to the i's finish.

Problems?

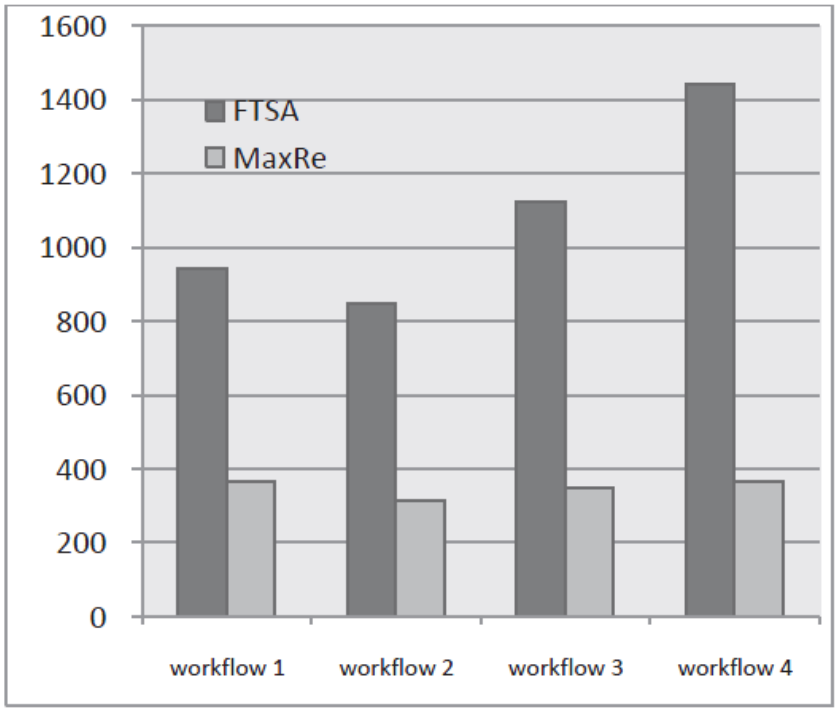


Fig. 1 Comparison on resource usage

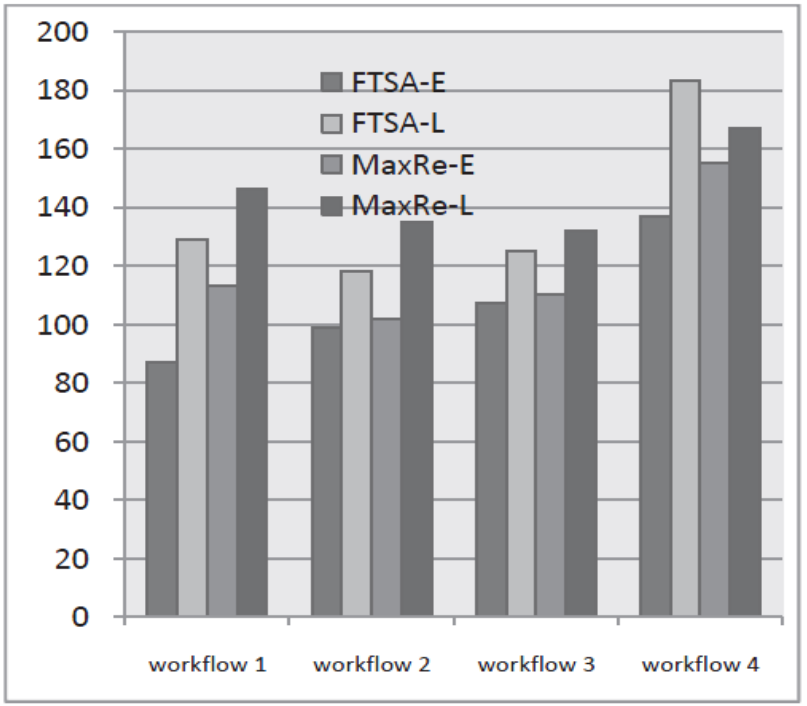


Fig. 2 Comparison on execution time

The MaxRe can save 70% resource to achieve corresponding reliability.
However, its performance on execution time is not that acceptable in certain situations.

Schedule minimum resources with considering the deadline and reliability constraints is the new challenge.

Outline

- 1. Previous work
  - Active replication scheme
  - MaxRe algorithm
  - Problems?
- 2. Before scheduling?
  - Subdeadline assignment
  - Reliability assignment
  - Task priority computation
- 3. Our proposal: DRR algorithm
- 4. Analysis & Experiments
- 5. Conclusion and future works

Subdeadline Assignment

Deadline requirement: *T.*

4 rules for subdeadline assignment in paper[1]:

P1: *The cumulative sub-deadline of any independent path between two synchronization tasks must be same.*

P2: *The cumulative sub-deadline of any path from entry to exit is equal to the overall deadline D .*

P3: *Any assigned sub-deadline must be greater than or equal to the minimum processing time of the corresponding task partition*

P4: *The overall deadline is divided over task partitions in proportion to their minimum processing time.*

2 rules for subdeadline assignment:

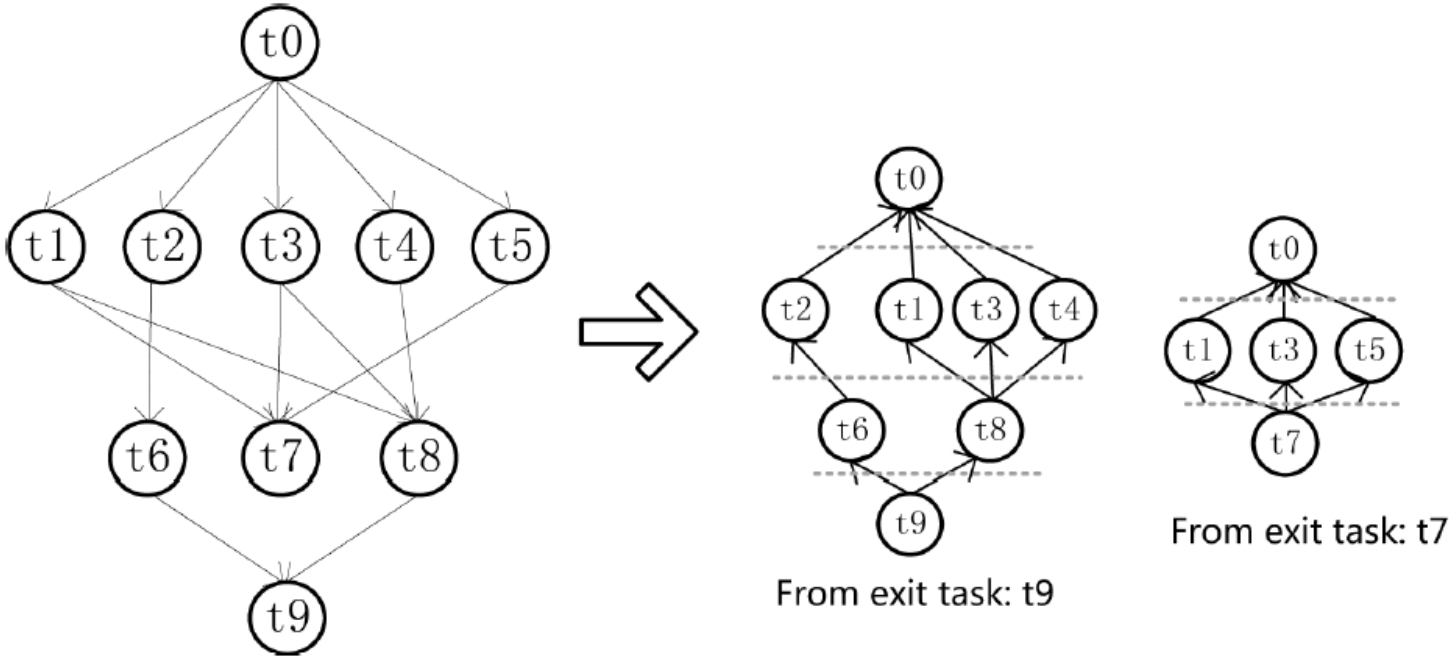
P1: The subdeadline can assure the overall deadline.

P2: The subdeadline for each task is not less than the finish time of each task.

[1] Y. Jia et al. e-Science and Grid Computing. 2005

Subdeadline Assignment

Breadth First Search based subdeadline assignment:



$$t = subdeadline[z] - \overline{C(e_{y,z}, l)} - \overline{ET\,(z, p)};$$
$$subdeadline[y] = min(t, subdealine[y]);$$

Where: ***C(e,l)*** is the communication cost between parent *y* and child *z*,
ET(z,p) is the execution time of task z.

Reliability Assignment

- Reliability requirement: *R*.
  - For each task, the subreliability should not less than: r=$\sqrt[n]{R}$ (n is the number of tasks).

- Reliability provided by the system, two parts:
  - Computation reliability
  - Communication reliability

Computation Reliability

Assumption:

- The arrival of failures follows a Poisson distribution:

$$f(k,\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- Let ***k=0***, the reliability value during period ***t*** is:

$$R = e^{-\lambda t}$$

Communication Reliability

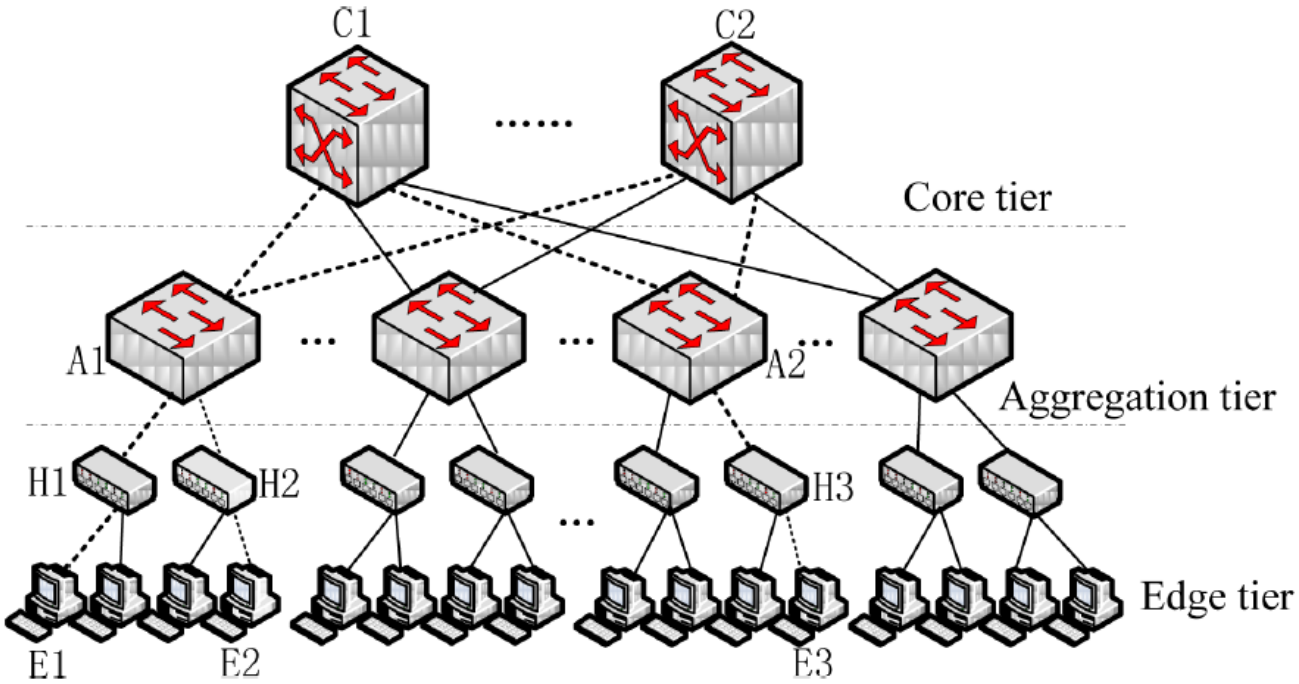


Figure 1. The three-tier network topology model

$$R_{comm}(E1, E2) = R_{E1-H1-A1} \cdot R_{A1-H2-E2}$$

$$R_{comm}(E1, E3) = R_{E1-H1-A1} \cdot R_{A2-H3-E3} \cdot R_{C1\ or\ C2}$$

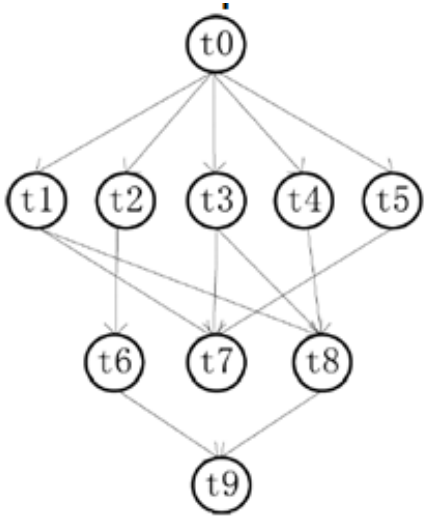
[1] M. Al-Fares et al., SIGCOMM2008

Task Priority Computation

- 2 available methods on computing the task priority:

$$bl\,(x) = \overline{ET\,(x,p)} + \max_{y \in child(x)} \left(\overline{C(e_{x,y},l)} + bl\,(y) \right)$$

$$tl(x) = \max_{z \in parent(x)} (tl(z) + \overline{ET(z,p)} + \overline{C(e_{z,x},l)})$$



  - HEFT[4], MaxRe[3] use the ***bl*** to compute the task priority
  - FTSA[2], CAFT[1], CPOP[4] use the ***tl + bl*** to compute the task priority.

[1] A. Benoit et al, Parallel computing,2009.
[2] A. Benoit et al, APDCM,2008.
[3] Laiping zhao et al, HPCC,2010.
[4] H. Topcuoglu et al, TPDS, 2002

Task Priority Computation

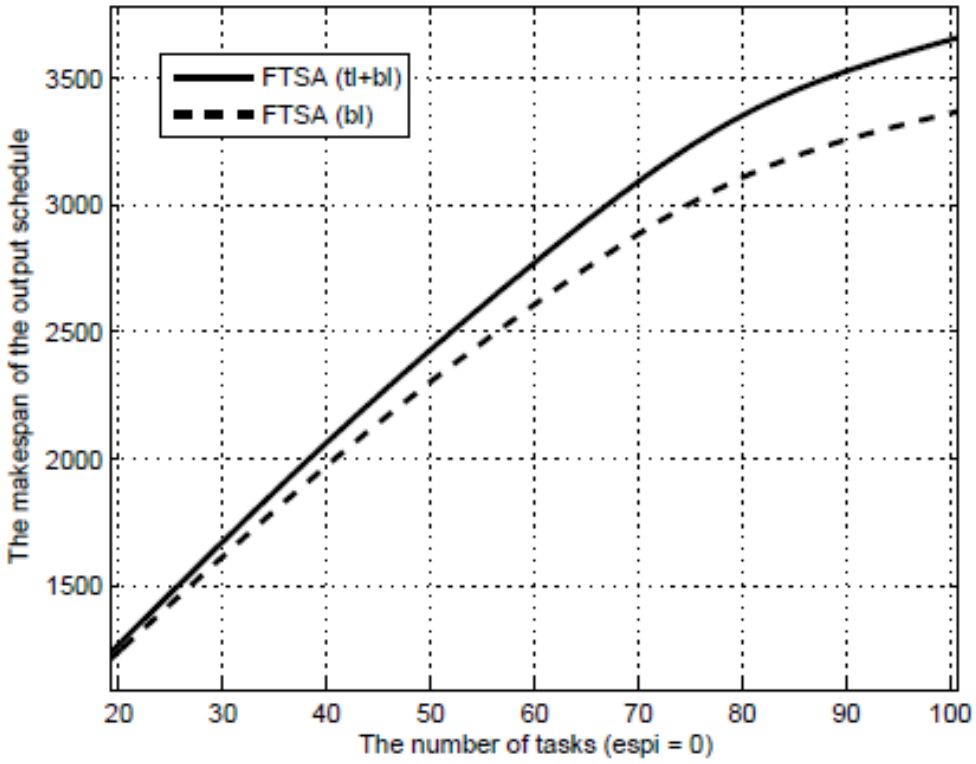


Figure 3. The comparison between FTSA(bl) and FTSA(tl+bl)

FTSA(bl) has a better performance than FTSA(tl+bl).
Therefore, we use the tl value in our scheduling

Outline

- 1. Previous work
  - Active replication scheme
  - MaxRe algorithm
  - Problems?
- 2. Before scheduling?
  - Subdeadline assignment
  - Reliability assignment
  - Task priority computation
- 3. Our proposal: DRR algorithm
- 4. Analysis & Experiments
- 5. Conclusion and future works

DRR Algorithm

- Deadline-Reliability-Resource-aware scheduling:
- Current Processor Execution Time (CPET):

$$CPET(x,p) = ET(x,p) + \sum_{y \in on(p)} ET(y,p)$$

$$\Downarrow$$

$$CPR(x,p) = e^{-\lambda \cdot CPET}$$

- Current Communication Time (CCT):

$$CCT_l(x,p) = C(e_{y,x},l) + \sum_{e \in on(l)} C(e,l)$$

$$\Downarrow$$

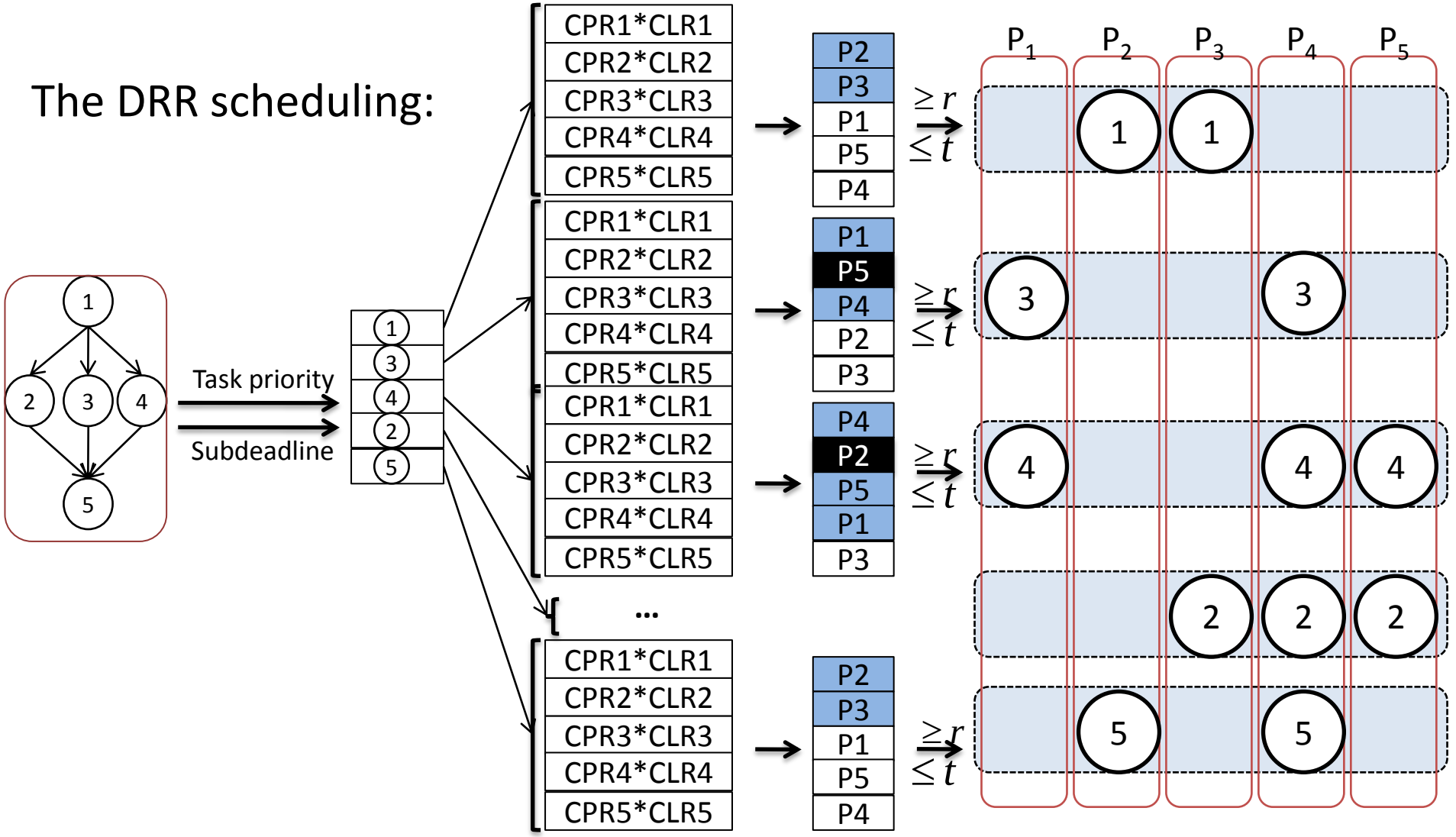
$$CLR_l(x,p) = e^{-\lambda \cdot CCT}$$

$$\Downarrow$$

$$CLR(x,p) = \prod_{y \in parent(x)} (1 - \prod_{r \in replica(y)} (1 - R_{comm}(p(r),p)))$$

DRR Scheduling

The DRR scheduling:



Analysis

Lemma 1:

*In the case of $\exists x \in V; FT(x) >$ subdeadline(x), the whole job **cannot** be successfully completed within the deadline constraint T.*

Lemma 2:

$\forall x \in$ entry nodes;ET(x; p) < subdeadline(x) is just the necessary condition of job's successful completing within deadline T.

Theorem 1:

In the case of the job being accepted in DRR, its output schedule can ensure the overall deadline T.

Analysis

Theorem 2:

Assume that for every task, the deserved number of replicas does not exceed n, and denote the reliability provided by DRR as $\mathcal{R}$, then $\mathcal{R} \geq R$.

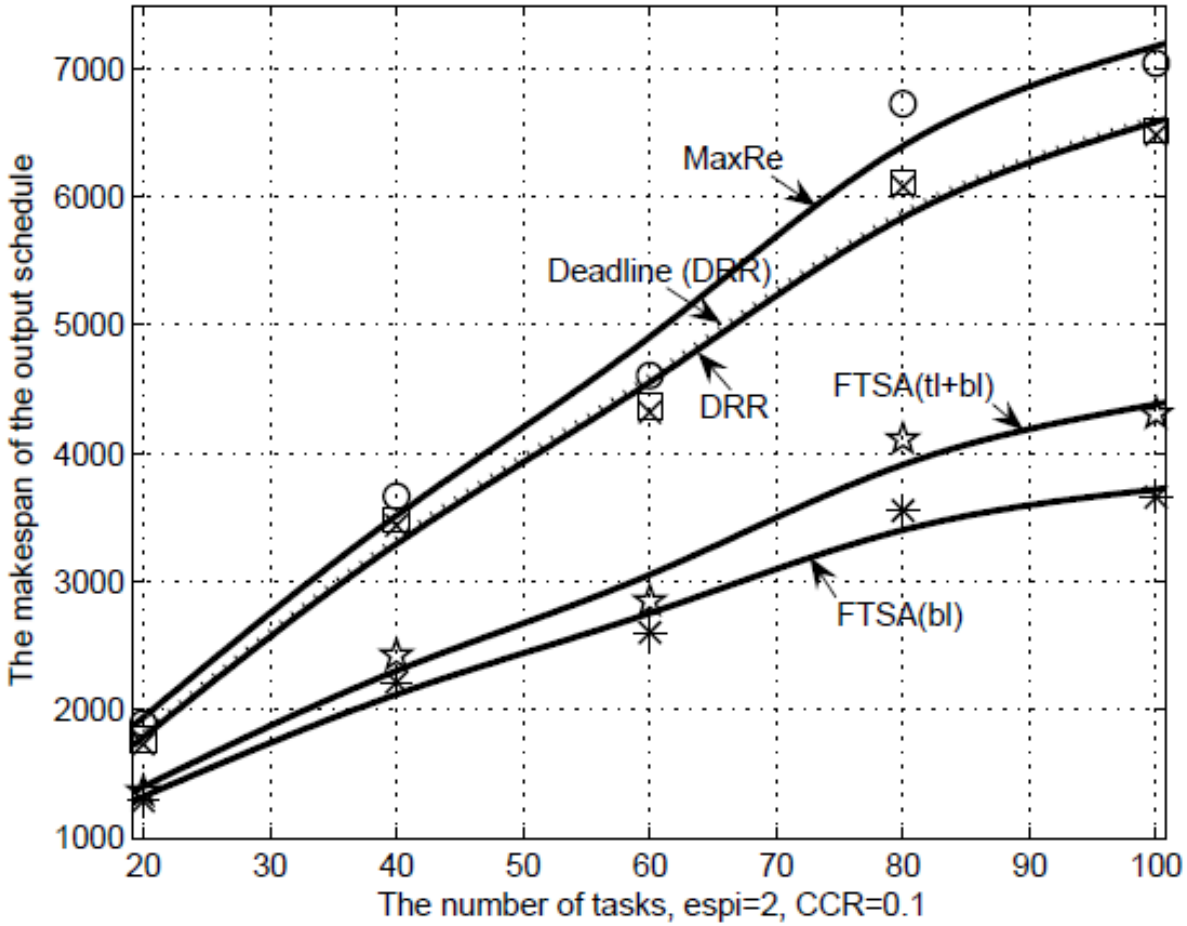
Theorem 3:

The time complexity of DRR is O(MN logM + N logN).

M is the number of processors, N denotes the number of tasks.

Experiments

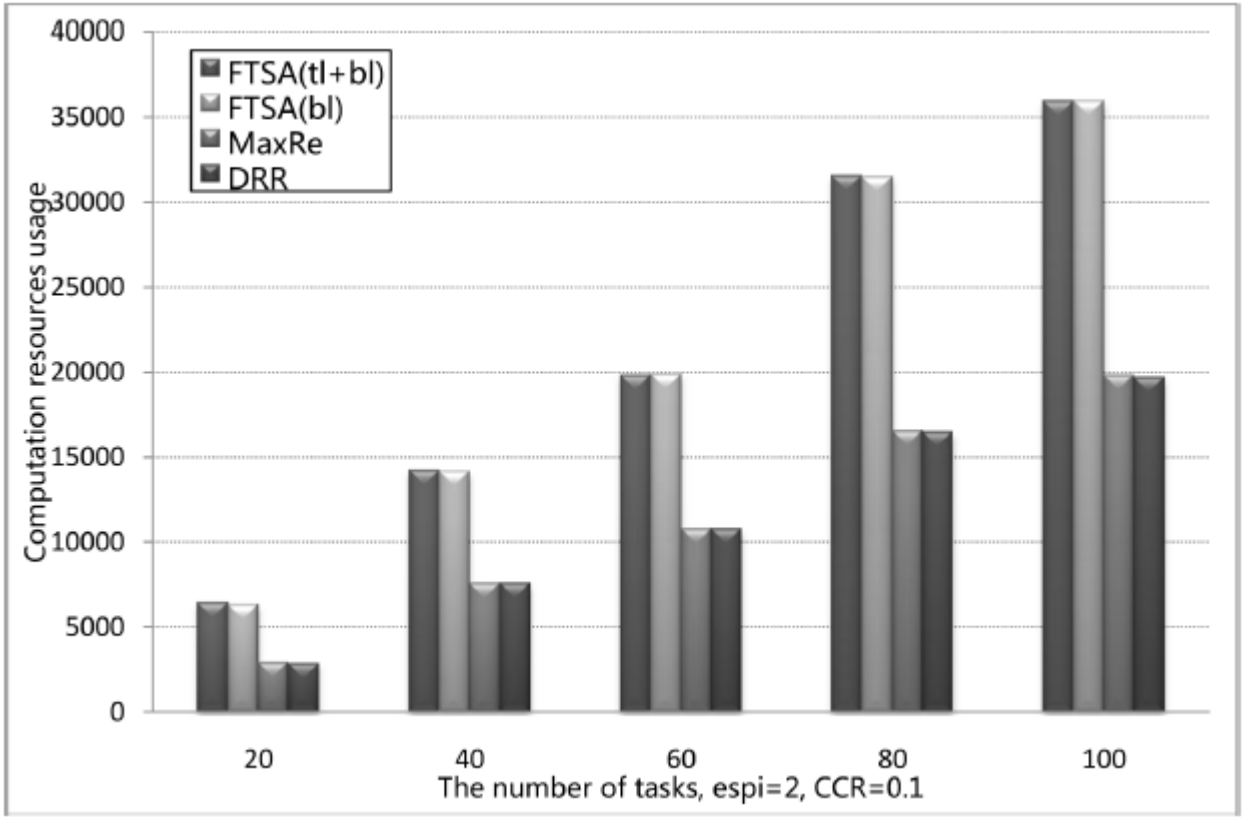
Makespan comparison:



DRR can finish the job within the deadline constraints.

Experiments

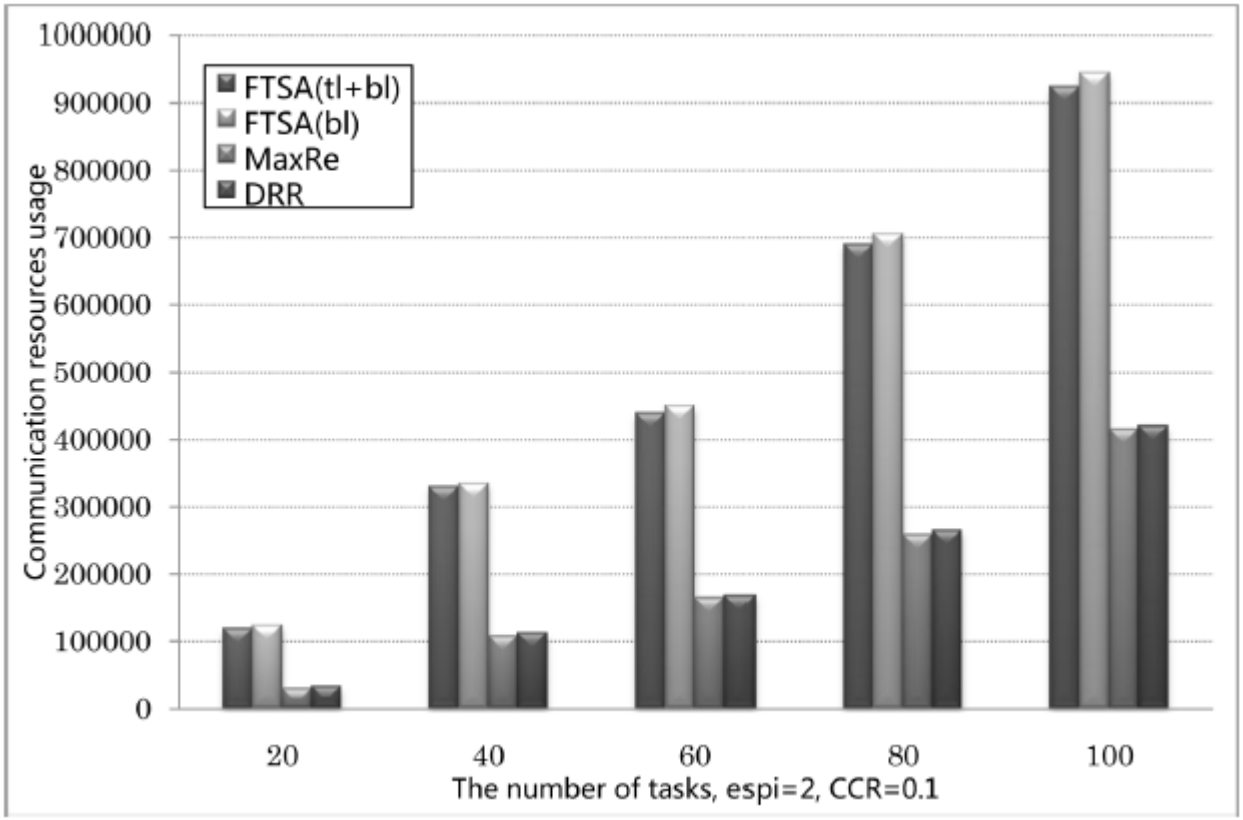
- Computation resource usage comparison:



DRR and MaxRe save 50% computation resources.

Experiments

- Communication resource comparison:



DRR and MaxRe save 70% communication resources.

Conclusion and future works

- This work improve the active replication scheme:
  - Achieves corresponding reliability with less resources;
  - Guarantee the deadline constraint.

- The future work:
  - Further reduce the resource usage with recovery.

- Thank you, Questions?