

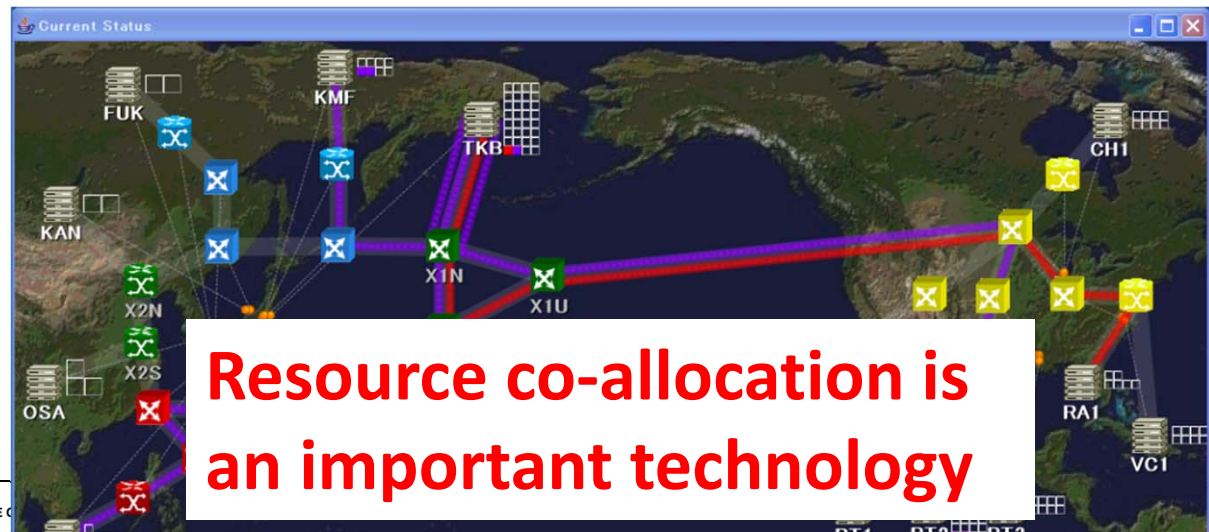
# An Advance Reservation-based Co-allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-guaranteed Grids

Atsuko Takefusa, Hidemoto Nakada,  
Tomohiro Kudoh, and Yoshio Tanaka

National Institute of Advanced Industrial Science and Technology  
(AIST)

# Resource Co-allocation for QoS-guaranteed Grids

- QoS is a key issue on Grid/Cloud
  - Network (=Internet) is shared by abundant users
- Network resource management technologies have enabled the construction of QoS-guaranteed Grids
  - Dynamic resource co-allocation demonstrated  
G-lambda and EnLIGHTened Computing [GLIF06,SC06]
  - Each network is dedicated and dynamically provisioned
  - No connectivity w/o reservation



# Preconditions of Our Co-allocation

- Commercial services
  - Some resources including network are provided by resource managers (RM) from commercial sectors
  - The resources will be charged
  - The RMs do not disclose all of resource information
- Advance reservation
  - Prediction-based scheduling systems, e.g. KOALA and QBETS, cannot guaranteed to activate co-allocated resources at the same time
    - The user has to pay for some commercial resources during the waiting time
- On-line reservation service
  - Try to complete resource co-allocation, quickly

# Issues for Resource Co-allocation for QoS-guaranteed Grids (1/2)

- Co-allocation of both computing and network resources
  - There are constraints between computers and the network links
  - Cannot use list scheduling-based approaches and network routing algorithms based on Dijkstra's algorithm, straightforwardly
- Reflecting scheduling options
  - Users: (a) reservation time, (b) price, and (c) quality/availability
  - Administrators: (A) load balancing among RMs, (B) preference allocation, and (C) user priority

# Issues for Resource Co-allocation for QoS-guaranteed Grids (2/2)

- Calculation time of resource co-allocation
  - Resource scheduling problems are known as NP-hard
  - Important to determine co-allocation plans with short calculation time, especially for on-line services

# Our Contribution

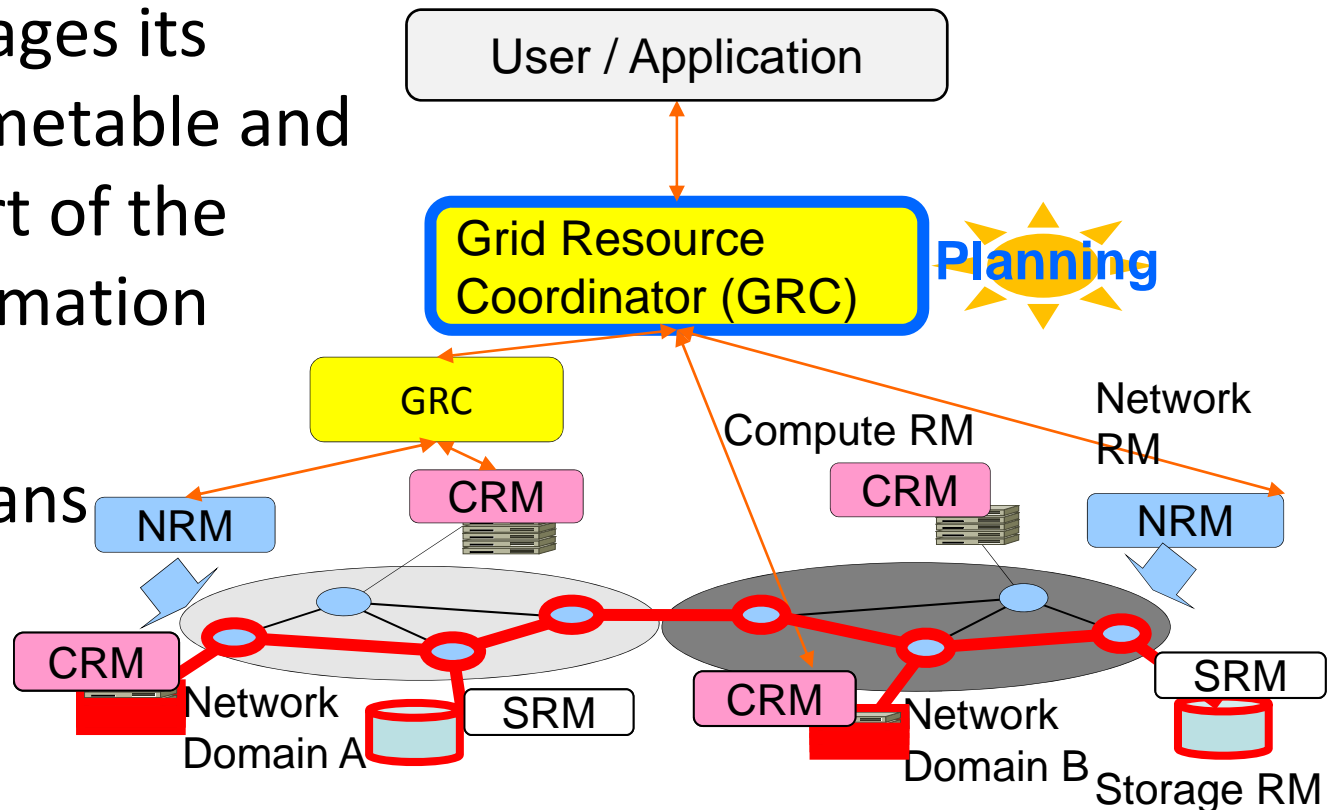
- Propose an on-line advance reservation-based co-allocation algorithm for distributed computers and network bandwidths
  - Model this resource co-allocation problem as an integer programming (IP) problem
  - Enable to apply the user and administrator options
- Evaluate the algorithm with extensive simulation, in terms of functionality and practicality
  - Can co-allocate both resources and can take the administrator options as a first step
  - Planning times using a general IP solver are acceptable for an on-line service

# The Rest of the Talk

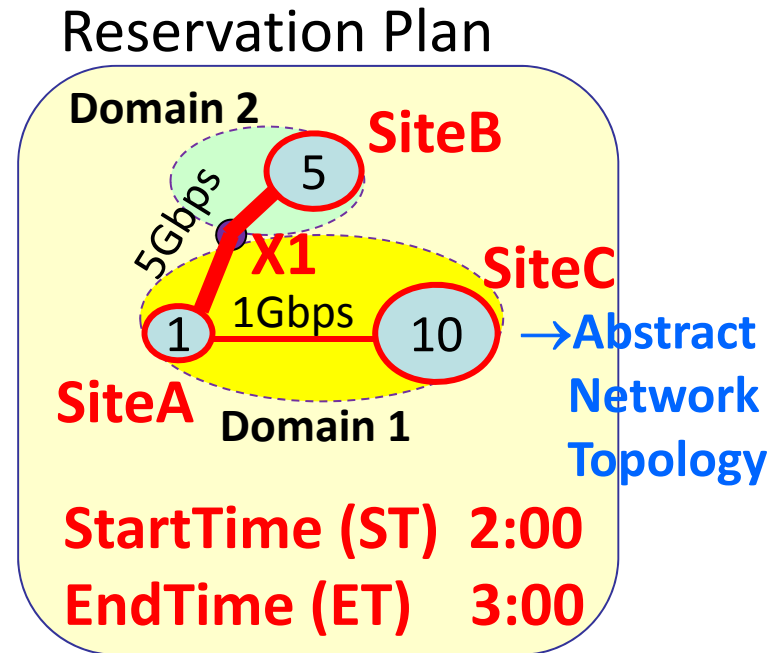
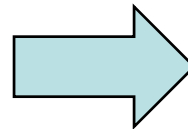
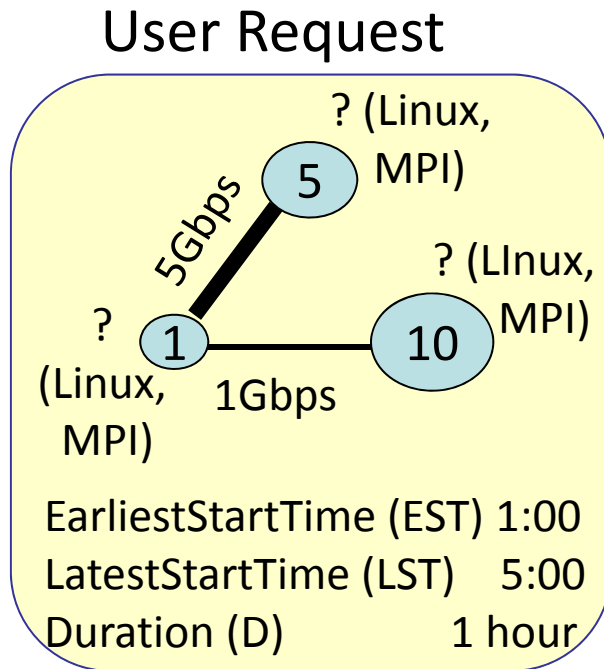
- Our on-line advance reservation-based co-allocation model
- An advance reservation-based co-allocation algorithm
  - Modeled as an IP problem
- Experiments on functional and practical issues
- Related work
- Conclusions and future work

# Our On-line Advance Reservation-based Co-allocation Model

- Consists of Global Resource Coordinators (**GRCs**) (= Grid scheduler) and resource managers (**RMs**)
- Each RM manages its reservation timetable and discloses a part of the resource information
- **GRC** creates reservation plans and allocates the resources



# User Request and Reservation Plan



## Resource Requirement Parameters

- Compute resources: # of CPUs/Cores, Attributes (e.g., OS)
- Network resources: Bandwidth, Latency, Attributes
- Time frames: Exact (ST and ET) or range (EST, LST, and D)

# The Steps of Resource Co-allocation

1. GRC receives a co-allocation request from User

2. **GRC Planner creates reservation plans**

2i. Selects N time frames from [EST, LST+D]

2ii. Retrieves available resource information at the N time frames from RMs

2iii. Determines N' ( $\leq N$ ) co-allocation plans using 2ii information

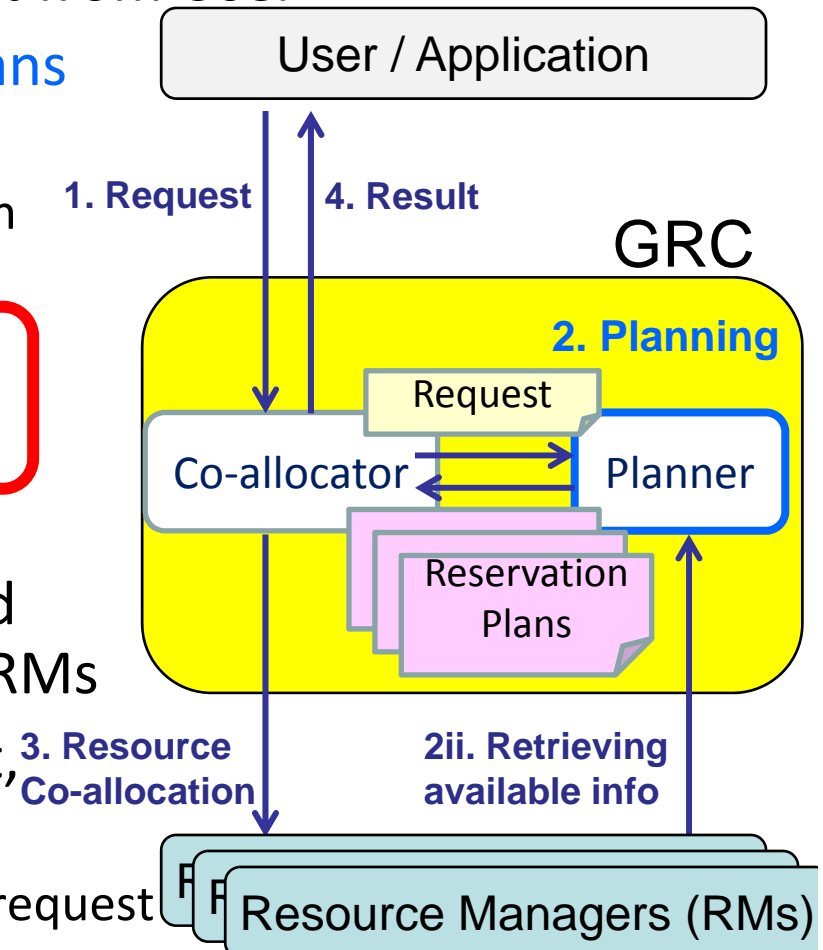
→ Modeled as an IP problem

2iv. Sorts N' plans by suitable order

3. GRC tries to co-allocate the selected resources in coordination with the RMs

4. GRC returns the co-allocation result, whether it has succeeded or failed

If failed, the User will resubmit an updated request



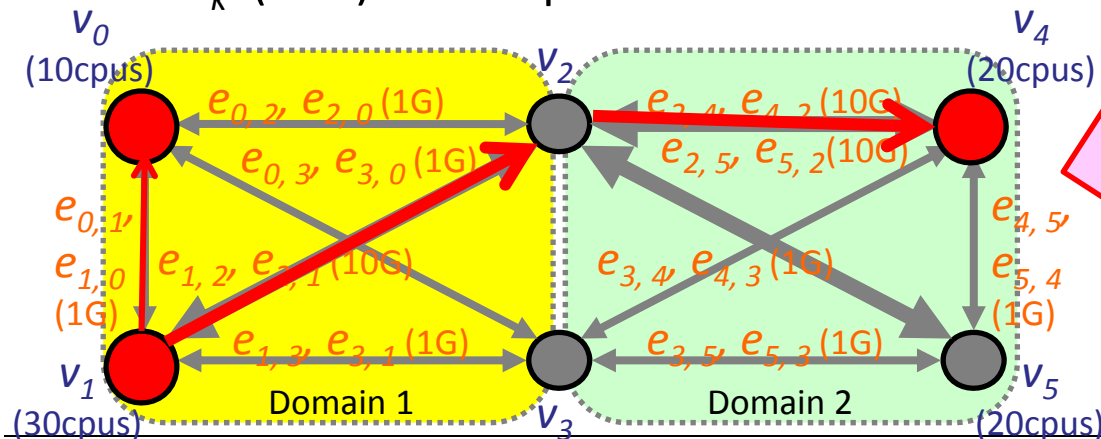
# Resource and Request Notations

## Resources: $G=(V, E)$

- $v_n (\in V)$  : Compute resource site or network domain exchange point
- $e_{o,p} (\in E)$  : Path from  $v_o$  to  $v_p$

## Resource parameters

- $wc_i (i \in V)$  : # of available CPUs
- $wb_k (k \in E)$  : Available bandwidth ( $e_{o,p}$  and  $e_{p,o}$  share the same  $wb_k$ )
- $vc_i (i \in V)$  : Value per unit of each CPU
- $vb_k (k \in E)$  : Value per unit of bandwidth

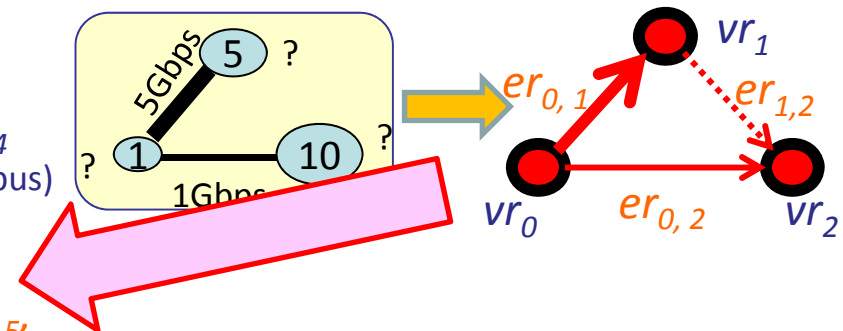


## Request: $G_r=(V_r, E_r)$

- $vr_m (\in V_r)$  : Requested compute site
- $er_{q,r} (\in E_r)$  : Requested network between  $vr_q$  and  $vr_r$

## Request parameters

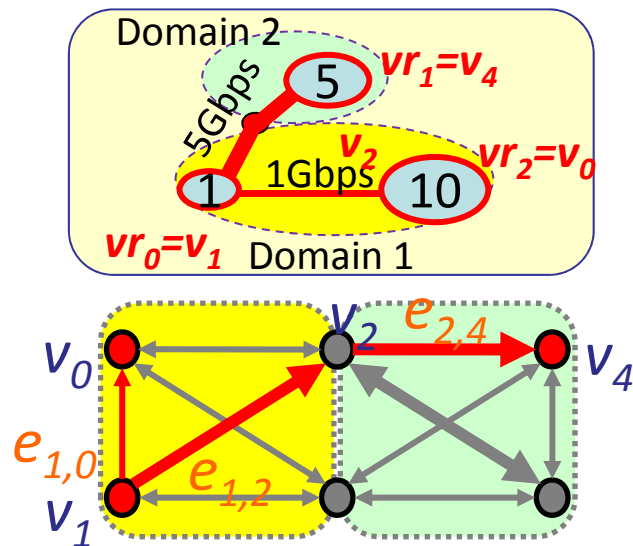
- $rc_j (j \in V_r)$  : Requested # of CPUs
- $rb_l (l \in E_r)$  : Requested bandwidth



# Modeling as a 0-1 IP Problem

$$X \text{ (Compute site plan)} = x_{i,j} \in \{0, 1\} \quad (i \in V, j \in V_r) \quad (1)$$

$$Y \text{ (Network path plan)} = y_{k,l} \in \{0, 1\} \\ (k=(m, n) \in E, m, n \in V, \\ l=(o, p) \in E_r, o, p \in V_r) \quad (2)$$



Request

$$X = \begin{matrix} \begin{matrix} vr_0 \\ vr_1 \\ vr_2 \end{matrix} & \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \text{Resources} \end{matrix}$$

$$Y = \begin{matrix} \begin{matrix} er_{0,1} \\ er_{0,2} \\ er_{1,2} \end{matrix} & \begin{matrix} e_{0,1} & e_{1,0} & e_{1,2} & e_{2,1} & e_{2,4} & e_{4,2} & \dots \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \end{bmatrix} \end{matrix}$$

# Objective function and Constraints

*Minimize*

$$\sum_{i \in V, j \in V_r} vc_i \cdot rc_j \cdot x_j + \sum_{k \in E, l \in E_r} vb_k \cdot rb_l \cdot y_{k,l} \quad (3)$$

*Subject to*

$$\forall j \in V_r, \sum_{i \in V} x_{i,j} = 1 \quad (4)$$

$$\forall i \in V, \sum_{j \in V_r} x_{i,j} \leq 1 \quad (5)$$

$$\forall i \in V, \sum_{j \in V_r} rc_j \cdot x_{i,j} \leq wc_i \quad (6)$$

$$\forall l \in E_r, \sum_{k \in E_r} y_{k,l} \begin{cases} \geq 1 & (rb_l \neq 0) \\ = 0 & (rb_l = 0) \end{cases} \quad (7)$$

$$\forall k \in E, \sum_{l \in E_r} rb_l \cdot y_{k,l} \leq wb_k \quad (8)$$

$$\forall l = (o, p) \in E_r, \forall m \in V, \sum_{n \in V, m \neq n} y^{(n,m), (o,p)} - \sum_{n \in V, m \neq n} y^{(m,n), (o,p)} = \begin{cases} x_{m,o} - x_{m,p} & (rb_l > 0) \\ 0 & (rb_l = 0) \end{cases} \quad (9)$$

(3) : Minimize the sum of resource values

(4),(5),(6) : Constraints on the compute site plan X

(4) Select 1 site for each requested site

(5) Each site is selected to 0/1 site

(6) Each selected site can provide requested # of CPUs

(7),(8) : Constraints on the path plan Y

(7) The sum of  $y_{k,l}$  becomes more than 1, if requested

(8) Each selected path can guarantee requested bandwidth

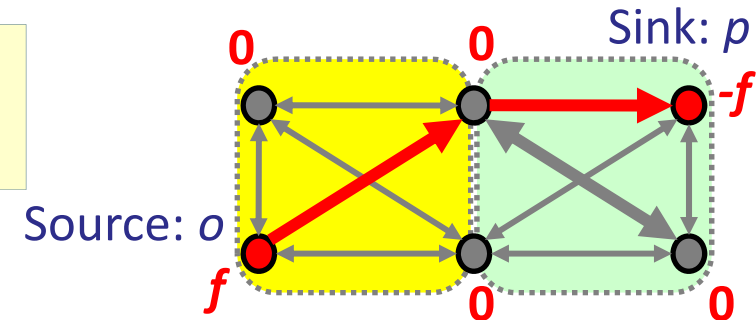
(9) : Constraint on both X and Y

# Application of Mass Balance Constraints

$$\forall l = (o, p) \in E_r, \forall m \in V,$$

$$\sum_{n \in V, m \neq n} y(n, m), (o, p) - \sum_{n \in V, m \neq n} y(m, n), (o, p) = \begin{cases} x_{m,o} - x_{m,p} & (rbl > 0) \\ 0 & (rbl = 0) \end{cases} \quad (9)$$

Sum of outflow Sum of inflow 0 / f / -f



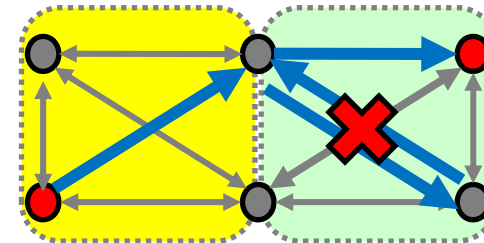
- Mass Balance Constraints (Kirchhoff's current law)
  - Except for source and sink, the sum of inflows equals to the sum of outflows
- Application of the constraints
  - Assume **requested network**  $l = (o, p) (\in E_r)$  is "current" from  $o$  to  $p$  and the flow = 1
    - Right-hand side becomes 0 / 1 / -1
  - $x_{m,o} = 1$  when  $m (\in V)$  is source, or  $x_{m,p} = 1$  when  $m$  is sink
    - Right-hand side could be represented as  $x_{m,o} - x_{m,p}$

# Additional Constraints

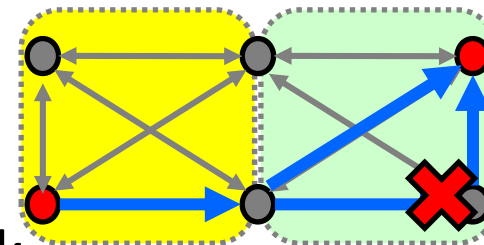
- Calculation times of 0-1 IP become exponentially long, due to NP-hard
- Propose additional constraints, which are expected to make calculation times shorter

Subject to

$$\forall l \in E_r, \forall m, n \in V (m \neq n), y_{(m,n),l} + y_{(n,m),l} \leq 1 \quad (10)$$



$$\forall l \in E_r, \sum_{k \in E} y_{k,l} \leq P_{\max} \quad (11)$$



Specifies  $P_{max}$ , the maximum of the number of paths for each network

→ Solutions might not be optimal

# Reflecting co-allocation options

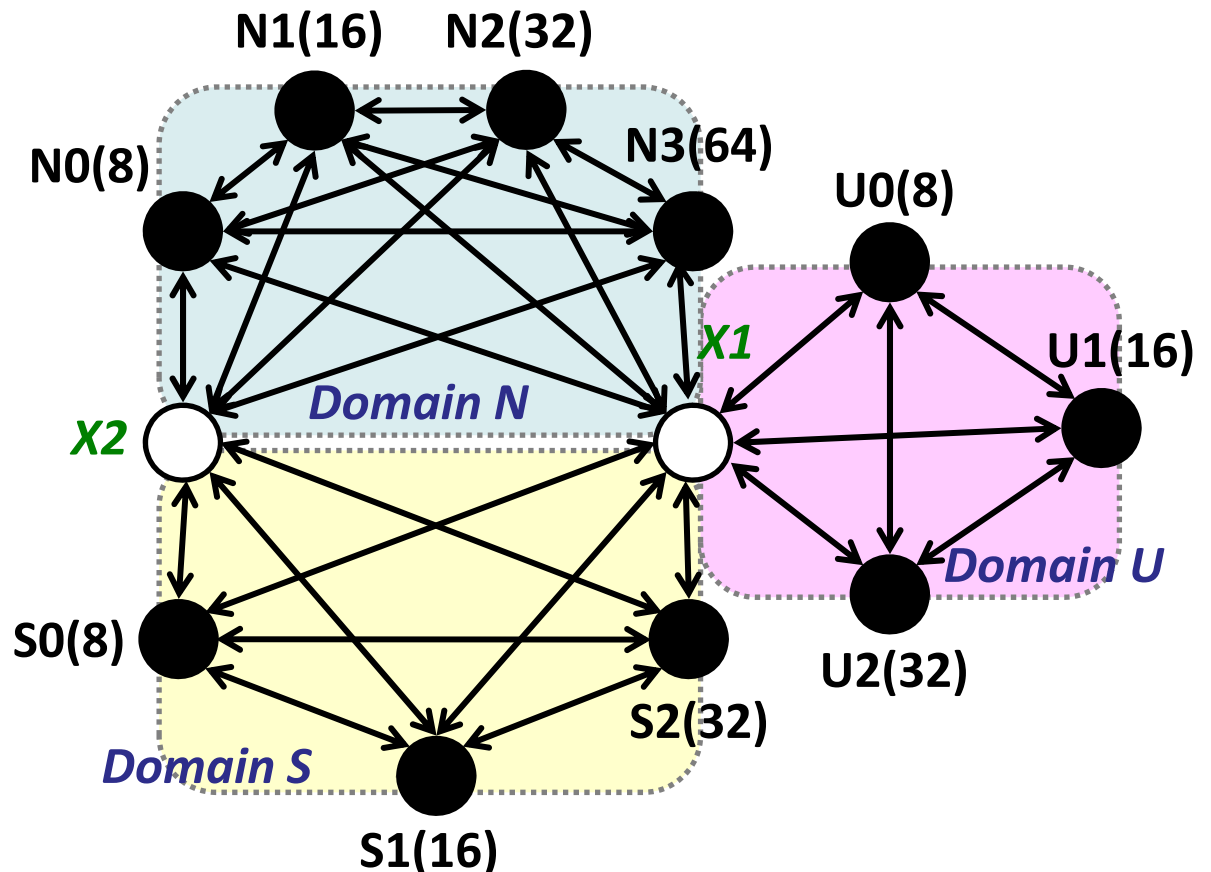
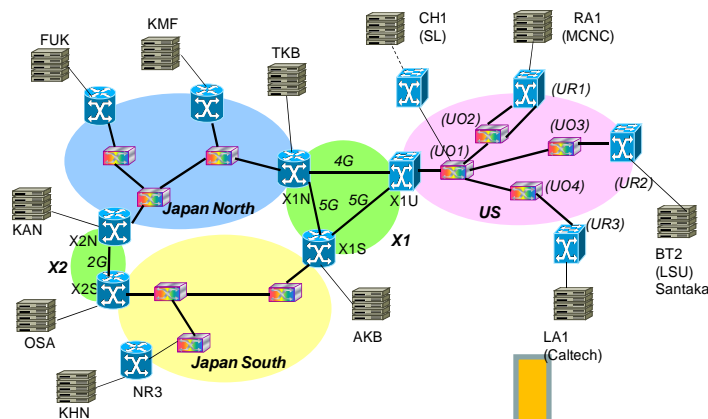
- User options
  - (a) Reservation time → Sort plans by times in stage 2iv
  - (b) Price → Sort plans by the total price
  - (c) Quality/Availability → Set  $vc_i$  and  $vb_k$  to their quality, modify the objective function, and then sort plans by the total value
- Administrator options
  - (A) Load balancing among RMs
  - (B) Preference allocation → Set  $vc_i$  and  $vb_k$  to weights of each resource
  - (C) User priority → Modify the retrieved available resource information

# Experiments

- Evaluate the algorithm with extensive simulation
  - Assume an actual international testbed
- Experiments on functional issues
  - Can co-allocate both compute and network resources
  - Can take the [administrator](#) options as a first step
- Experiments on practical issues
  - Compare planning times using additional constraints and different IP solvers
  - Planning times are acceptable for an on-line service

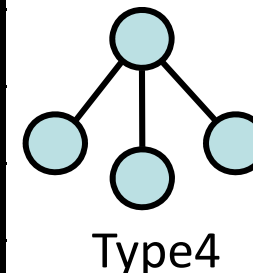
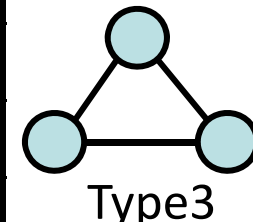
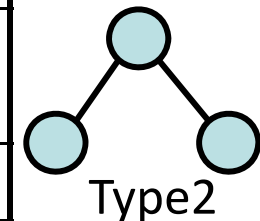
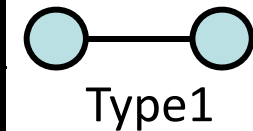
# Experimental Environment

- Assume an actual testbed used in the G-lambda and EnLIGHTened Computing experiments
  - 3 network domains, 2 domain exchange points, and 10 sites



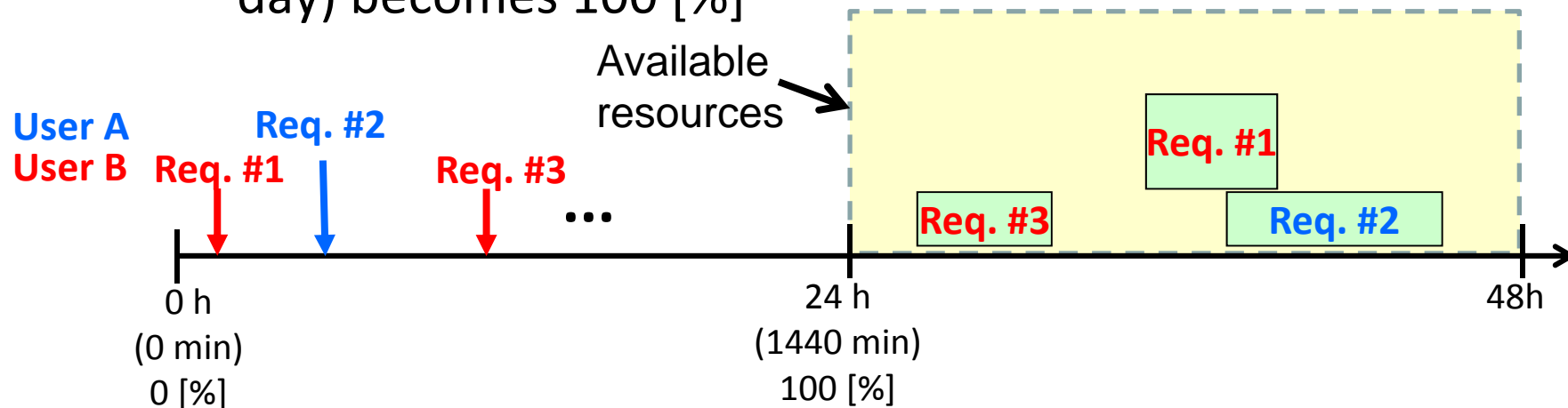
# Simulation Settings

Environment Settings	
Comfiguration	GRM=1, NRM=3, CRM=10
# of sites / Domain	4 / N, 3 / S, 3 / U
Domain exchange points	X1{N, S,U}, X2{N, S}
# of CPUs / CPU unit value	N{8, 16, 32, 64}, S{8, 16, 32}, U {8, 16, 32} / 1
Bandwidth [Gbps] / unit value	in-domain : 5 / 5, inter domain : 10 / 3
Resource Requirement Settings	
Users	UserA, UserB
Resource requirement types	Type 1, 2, 3, 4 (Uniform distribution) →
Requested # of CPUs	1, 2, 4, 8 for all sites in all types (Uniform)
Requested bandwidth	1 [Gbps] for all paths in all types
Interval of each user request	Poisson arrivals
Reservation duration (D)	30, 60, 120 [min] (Uniform distribution)
LST - EST + D	D × 3



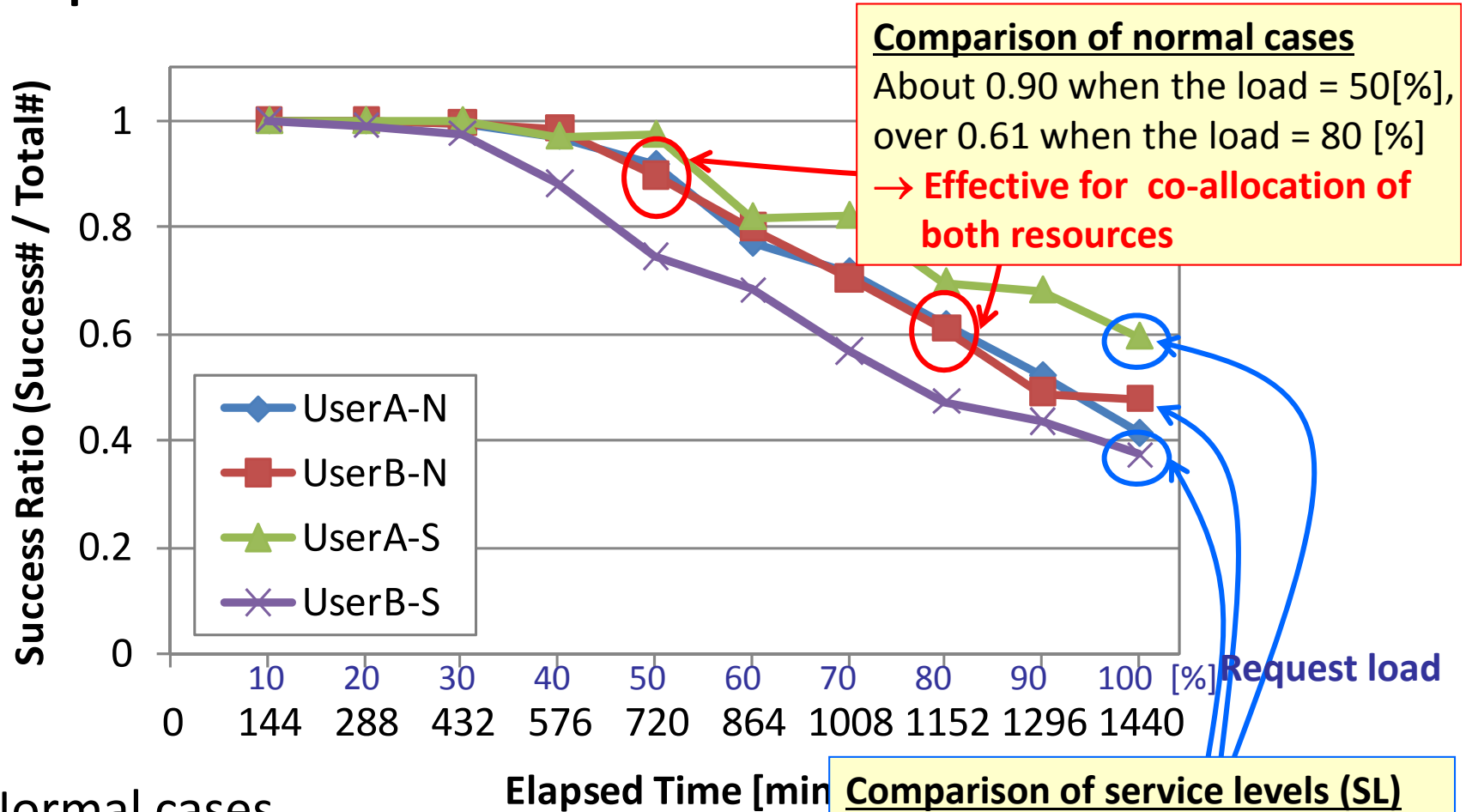
# Simulation Scenarios

- In the first 24 hours, each user sends co-allocation requests for the next 24 hour resources
  - The request load (= Ideal resource utilization on the next day) becomes 100 [%]



- # of reservation plans  $N = 10$ 
  - Time frames are selected equidistantly

# Comparison of Co-allocation Success Ratios



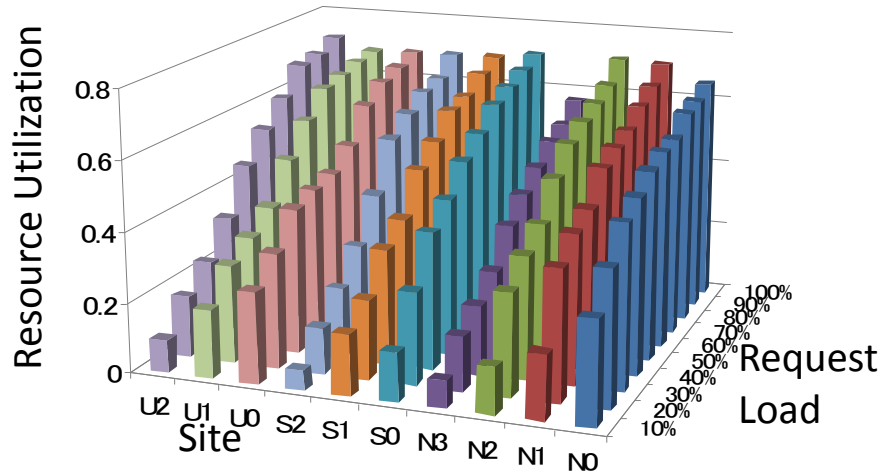
**Comparison of normal cases**  
 About 0.90 when the load = 50[%],  
 over 0.61 when the load = 80 [%]  
 → Effective for co-allocation of both resources

**Comparison of service levels (SL)**  
 -N : UserA and B are comparable  
 -S : UserA is 0.60 and UserB is 0.37 when the load = 100[%]  
 → Can take option (C) User priority

- N : Normal cases
- S : Configured different service levels
  - UserB is set to a low priority
  - For each UserB request, # of available resources is reduced to half amount

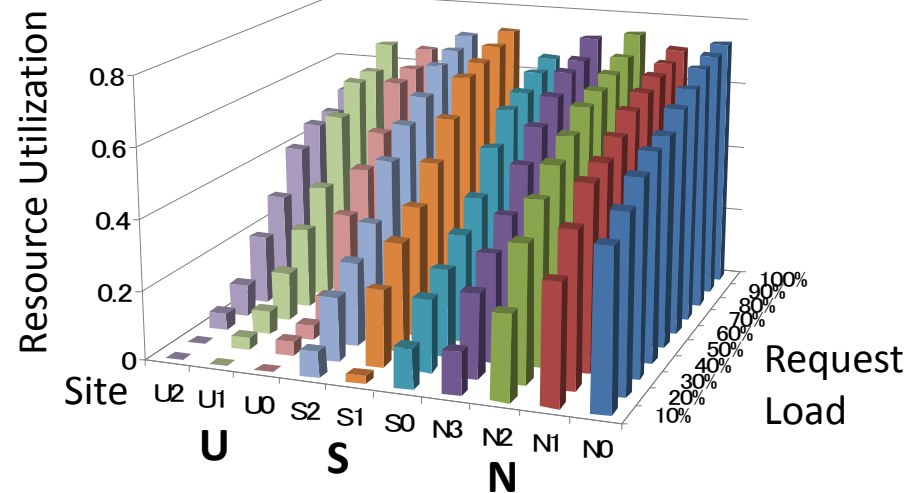
# Comparison of Resource Utilizations with administrator options (A) and (B)

(A) Load balancing



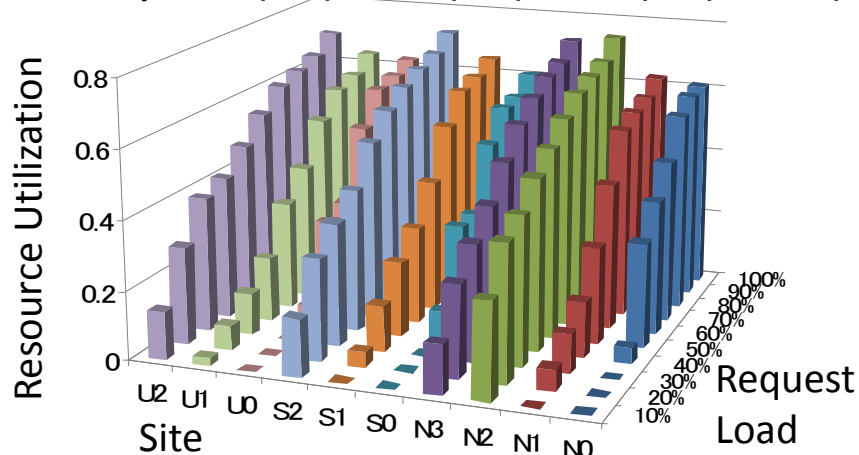
(B1) Preference allocation by domain

(Priority : N > S > U)



(B2) Preference allocation by # of CPUs

(Priority : \*3(64) > \*2(32) > \*1(16) > \*0(8))



$vc_i$  is set as follows:

(B1)  $N^* = 1, S^* = 10, U^* = 100$

(B2)  $*3 = 1, *2 = 10, *1 = 100, *0 = 1000$

Preferred resources are selected first

→ Can take options (A) and (B)

# Experiments on practical issues

- Investigate if planning times are acceptable for an on-line service
- Compare planning times using
  - Additional constraints
  - Different IP solvers
- IP solvers
  - General IP solver: [GLPK](#) (free, but slow)
  - SAT-based solver: [MiniSat](#) and [Sugar++](#)
    - Sugar++ enables a SAT solver to solve IP problems
- Experimental settings
  - CPU: Intel Core2 Quad Q9550 (2.83GHz),
  - OS: CentOS 5.0, kernel 2.6.18 x86\_64, Memory: 4GB

# Comparison of planning times

Additional constraint are effective  
Acceptable for an on-line service

MiniSat-st-1 is the best performance

Solver - constraint	Avg. [sec]	Max. [sec]	$\sigma$
GLPK	0.779	8.492	1.721
GLPK-st	0.333	4.205	0.700
MiniSat-st	12.848	216.434	27.914
MiniSat-st-1	1.918	2.753	0.420

GLPK is dominant

→ IP solvers are suitable for our algorithm

"-st" : Additional constraints ( $P_{max}=2$ )

"-1" : # of SAT executions = 1 (Select only one satisfied solution)

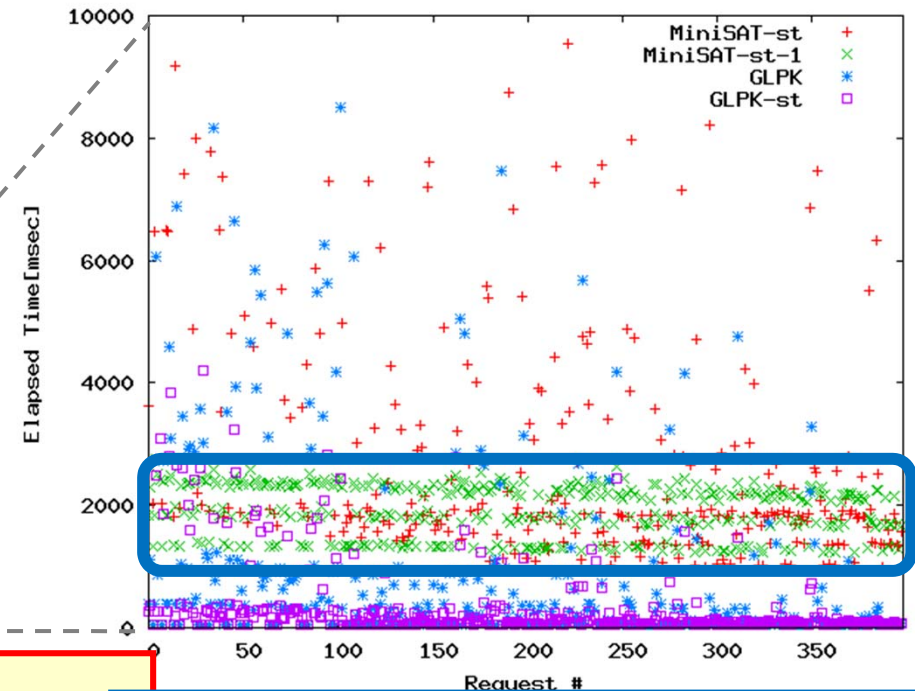
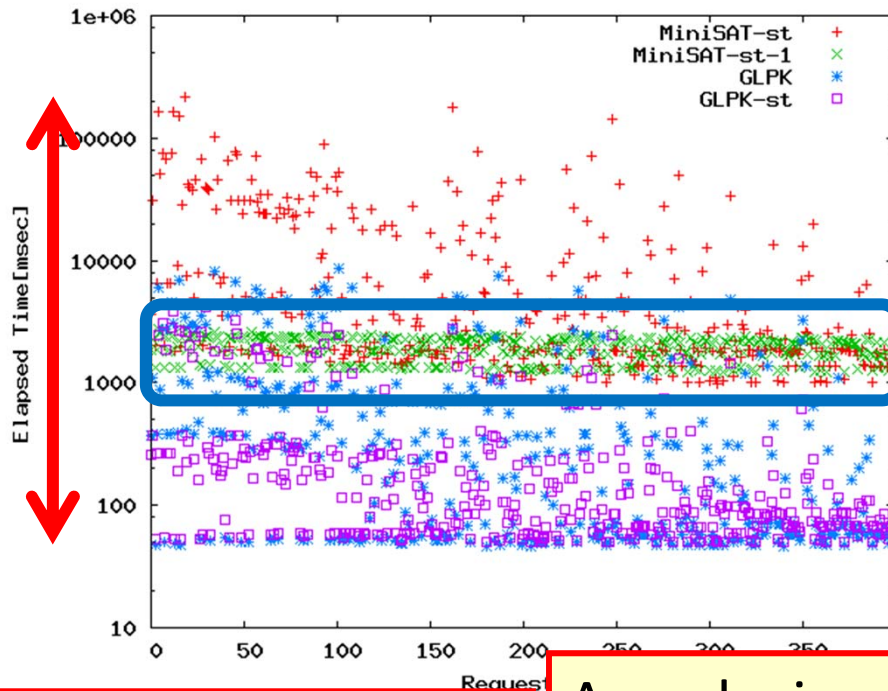
Quality of plans :  $GLPK \geq GLPK-st = MiniSat-st > MiniSat-st-1$

# Comparison of the Avg. planning times for each request

Planning times in log scale

Planning times from 0 to 10 [sec]

GLPK / GLPK-st / MiniSat-st / MiniSat-st-1



While MiniSat-st-1 is stable, the others are dispersed

As reducing the available resources planning times are decreasing

The results of MiniSat-st-1 are proportional to the # of sites in the requirement Types

# Discussion

- The coverage of IP problems is expanding
    - Performance of recent computers and the improvement of IP solvers
    - IP calculation times can be reduced by applying suitable constraints and approximate solutions
  - Our resource co-allocation model
    - The search area of a single GRC can be localized, because GRCs can consist hierarchically
    - The # of variables scales by the # of "computer sites", not "computers"
    - In practical use, additional constraints will increase, e.g., latency, execution environment, and required data locations
- Modeling as an IP problem is effective for our model

# Related Work

They cannot select suitable resources because the first found resources are selected

- Backtrack-based scheduling algorithm [Ando, Aida. 2007]
  - Enables both co-allocation and workflow scheduling
  - Co-allocation times become long and lots of resources are blocked, when the scheduler allocates resources incrementally
- Co-allocation algorithm for NorduGrid [Elmroth, Tordsson, 2009]
  - Search (1) computer sites and (2) paths between the selected sites, sliding a reservation time frame
  - Resource constraints make planning times long
- Co-reservation algorithm based on an optimization problem [Röblitz, 2008]
  - Network model is simple
  - Use all of resource information

**No algorithm can take co-allocation options**

# Conclusions

- We propose an **on-line advance reservation-based co-allocation algorithm for compute and network resources**
  - Modeled as an **integer programming (IP)** problem
  - Enable to apply the user and administrator options
- The experiments showed
  - Our algorithm can co-allocate both resources and can take the administrator options
  - Planning times using a general IP solver with additional constraints are acceptable for an on-line service

# Future Work

- Improve our algorithm and conduct further experiments on the scalability
- Apply sophisticated SLA and economy models and confirm that our algorithm can also take user options

# Acknowledgements

- Prof. Naoyuki Tamura and Mr. Tomoya Tanjo from Kobe University
- Prof. Katsuki Fujisawa and Mr. Yuichiro Yasui from Chuo University
- This work was partly funded by KAKENHI 21700047 and the National Institute of Information and Communications Technology