

# 多種資源を対象とする オンラインコアロケーション手法

竹房あつ子, 中田秀基,

工藤知宏, 田中良夫

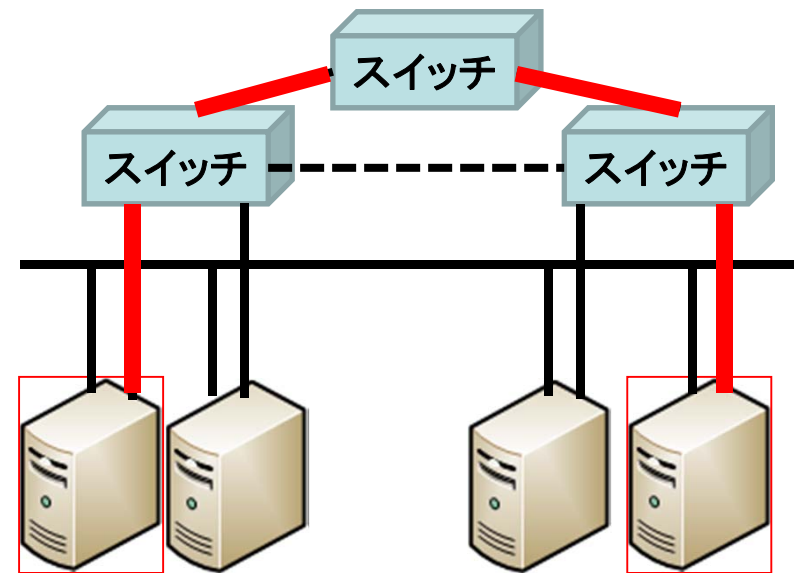
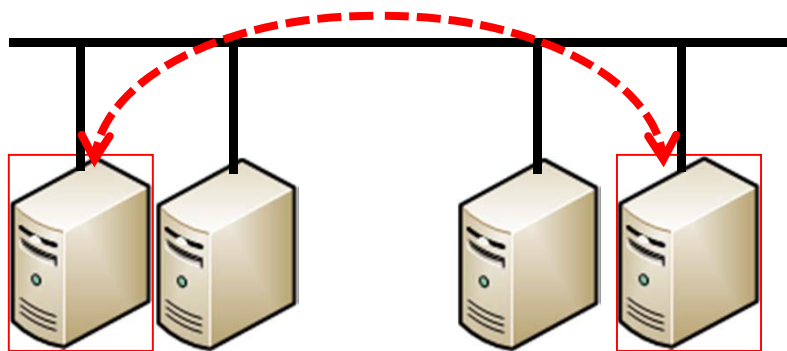
産業技術総合研究所

# 大規模データインテンシブ計算のための 資源管理

- 大規模データインテンシブ計算
  - 高エネルギー物理学, 地球科学, 天文学, センサーネットワーク等, 多分野で必要
  - 分散する複数組織の資源間で大規模データを共有, 処理
  - ネットワーク性能が処理性能に影響
- 計算機, ストレージとダイナミックサーキットネットワークを  
組み合わせた**仮想インフラ**構築が重要
  - G-lambda, EnLIGHTened, Internet2, GEANT
  - ネットワークの仮想化, 動的管理技術の発展

# ダイナミックサーキットネットワーク

- 通常の分散処理
  - ネットワークを共有
  - ベストエフォート
- ダイナミックサーキットネットワークを用いた分散処理
  - スイッチ／ルータで動的にパスを構成
  - 帯域保証が可能



# 仮想インフラ構築の課題

- ネットワークのトポロジを考慮した多種資源の同時割り当て(コアロケーション)が重要
- 既発表コアロケーション手法
  - バックトラック法で深さ優先探索[安藤, 合田, 2007]
  - 計算資源の選択→ネットワークの探索 [Elmroth, Tordsson, 2009]
  - 0-1整数計画法(0-1 IP)にモデル化[竹房, 2009]
  - 計算機とネットワークの割り当てのみ
  - 多種資源の柔軟な割り当てが必要
    - ストレージ(容量, I/O性能), ライブラリ, OS等属性情報
    - ユーザ/アプリが資源の一部を指定
      - 利用するデータの所在はユーザ/アプリが知っている

# 本研究の成果

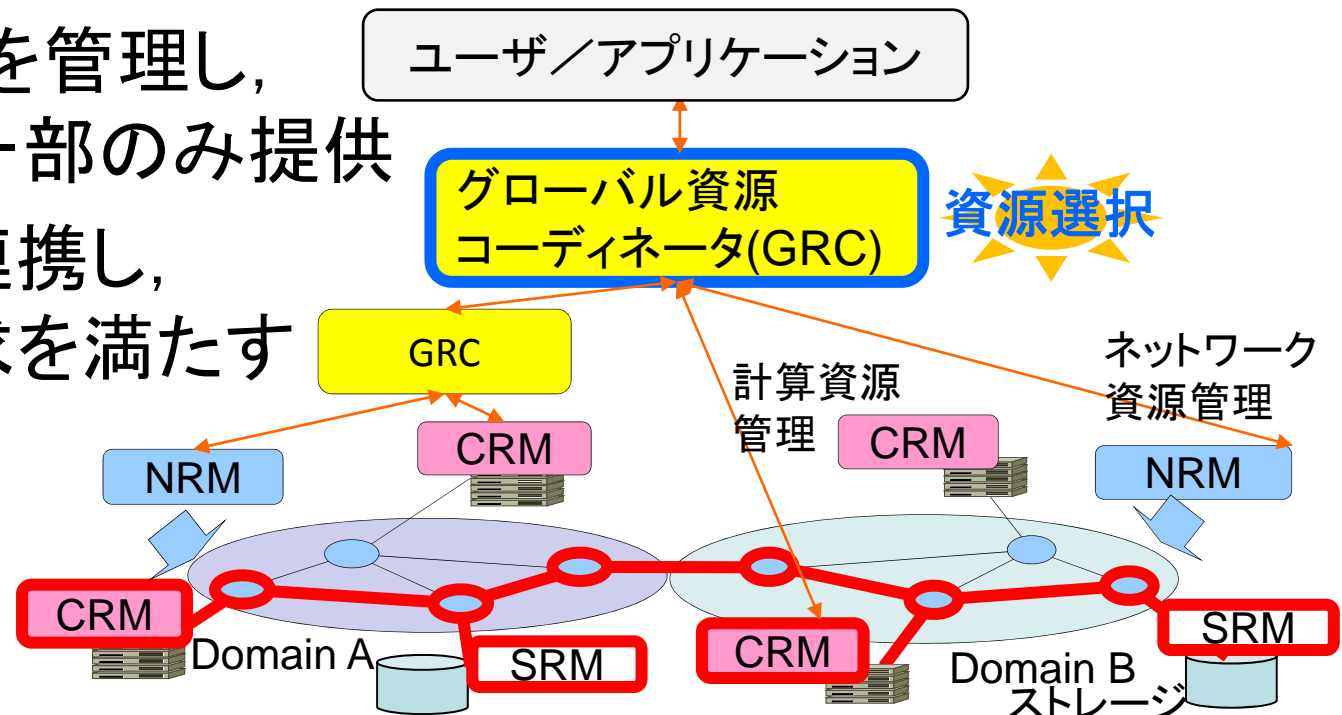
- 多種資源とその属性情報を考慮したオンラインコアロケーション手法を提案
  - 0-1 IPにモデル化した手法を改良
  - ユーザの資源指定にも対応
  - JavaLPインタフェースを用い、複数のソルバを適用可能
- 基本性能の評価
  - 10サイト3ネットワークドメインを想定し、求解時間を比較
  - 資源の一部が指定される場合は高速に処理可能
  - 属性指定がある場合は通常と同程度に処理

# 発表概要

- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- 基本性能の評価
- まとめ

# 前提とするコアロケーションモデル

- 事前予約. オンライン処理(↔バッチ)
- 各資源は複数の組織が提供
- 各資源の資源マネージャ(RM)とグローバル資源コーディネータ(GRC)で構成
- RMが予約表を管理し、資源情報は一部のみ提供
- GRCがRMと連携し、ユーザの要求を満たす資源を選択

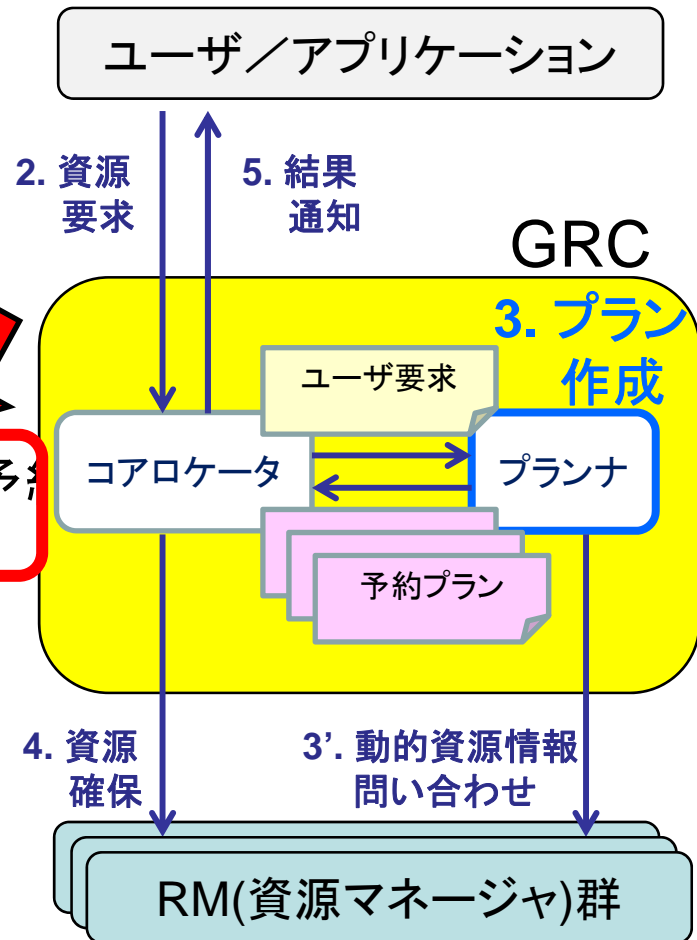
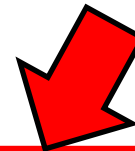


# 発表概要

- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- 基本性能の評価
- まとめ

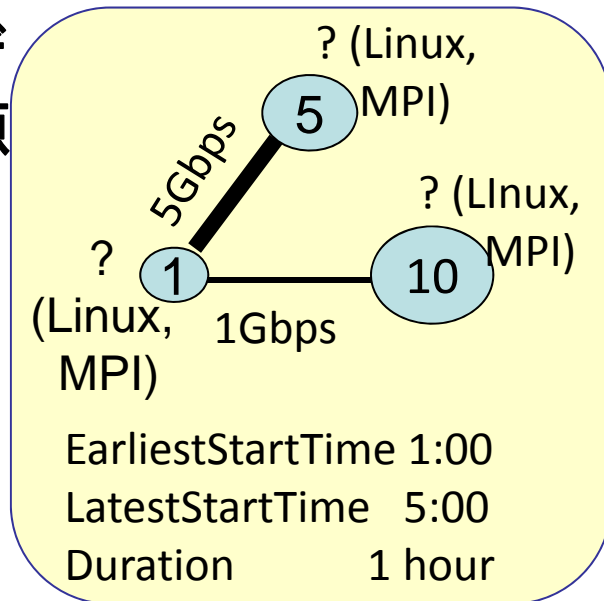
# コアロケーションの手順

1. 静的資源情報をあらかじめ取得
2. ユーザの資源要求を受け取る
3. **GRCのプランナが複数予約プラン作成**
  - 3a. 指定した時間帯から予約時間帯候補をN個選ぶ
  - 3b. 関連するRMに対し、N個の時間帯の動的情報を取得
  - 3c. 3bの情報から、**0-1 IPモデル**で、 $n(\leq N)$ 個の予約プランを作成
  - 3d.  $n$ 個の予約プランの優先順位を決定
4. 予約プランをもとに資源確保
5. 確保成功 / 失敗をユーザに通知  
失敗の場合、ユーザは条件を変えて再度要求

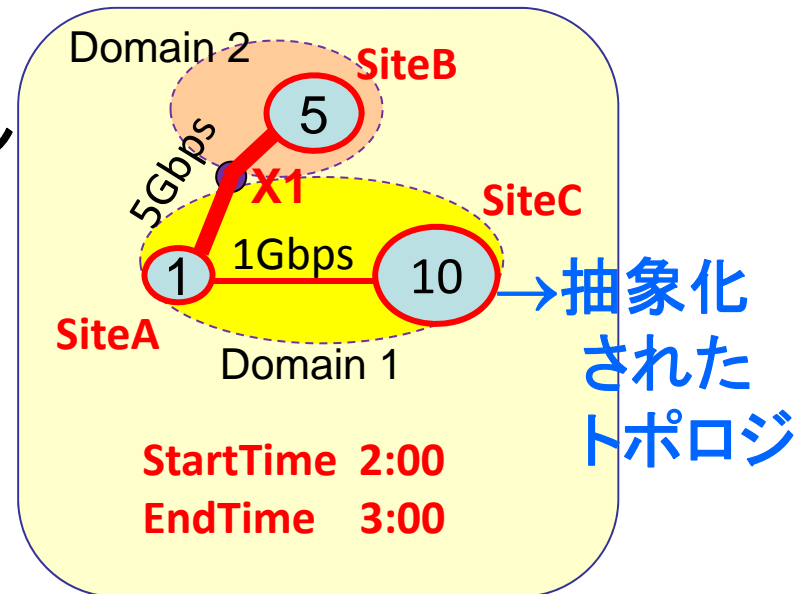
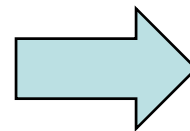


# ユーザの資源要求と予約プラン

ユーザ  
の資源  
要求



予約  
プラン



- 資源要求パラメータ

- 計算資源: CPU/コア数(以降CPU数), 属性情報(OS)
- ネットワーク資源: 帯域, 遅延, 属性情報
- 時刻: 直接指定 / 時間帯指定(→予約時間帯候補を選択)

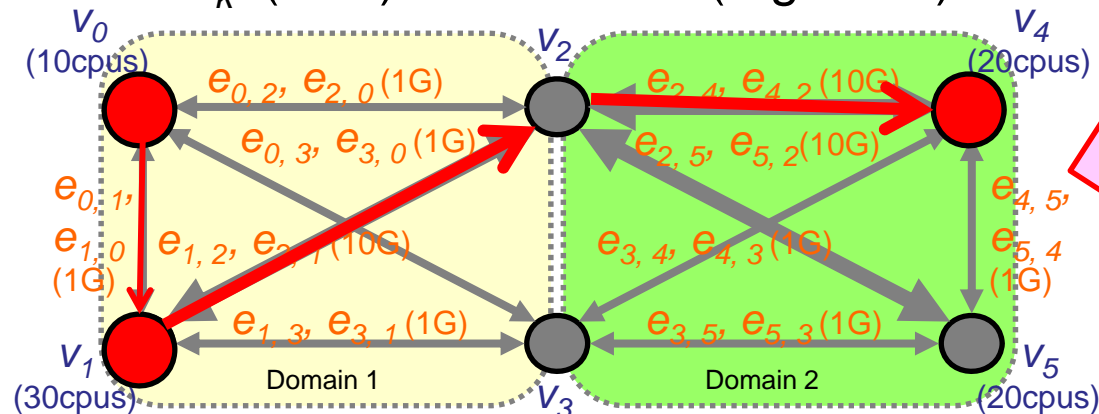
# 資源群と資源要求のモデリング

## 資源群: $G=(V, E)$

- $V$ :  $G$ の点の集合,  $E$ :  $G$ の枝の集合
- $v_q$ : 計算資源サイトまたはネットワークドメイン交換点
- $e_{o,p}$ : 資源または交換点間のパス ( $o$ : 始点,  $p$ : 終点)

## 資源パラメータ (枝は双方向で共有)

- $wc_i$  ( $i \in V$ ): CPU数,  $wb_k$  ( $k \in E$ ): 帯域
- $vc_i$  ( $i \in V$ ): CPU数の重み(e.g. 価格),  $vb_k$  ( $k \in E$ ): 帯域の重み(e.g. 価格)

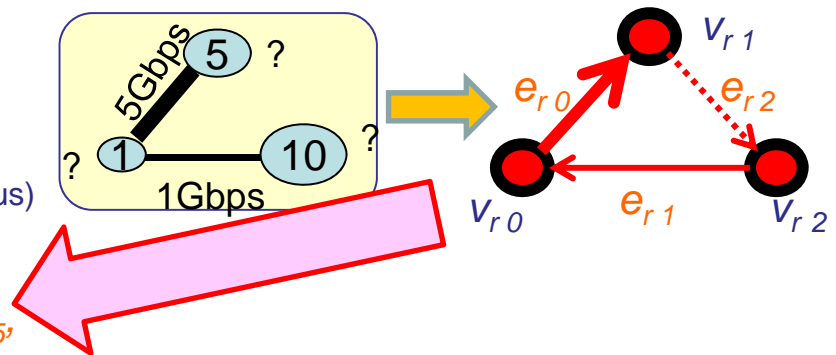


## 資源要求: $G_r=(V_r, E_r)$

- $V_r$ : 要求資源サイトの集合
- $E_r$ :  $V_r$ 間を結ぶ枝の集合

## 資源要求パラメータ

- $rc_j$  ( $j \in V_r$ ):  $V_r$ に必要なCPU数
- $rb_l$  ( $l \in E_r$ ):  $E_r$ に必要な帯域





# 目的関数と制約条件

## 目的関数

Minimize

$$\sum_{i \in V, j \in V_r} v c_i \cdot r c_j \cdot x_j + \sum_{k \in E, l \in E_r} v b_k \cdot r b_l \cdot y_{k,l} \quad (3)$$

## 制約条件

Subject to

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

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

$$\forall i \in V, \sum_{j \in V_r} r c_j \cdot x_{i,j} \leq w c_i \quad (6)$$

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

$$\forall k \in E, \sum_{l \in E_r} r b_l \cdot y_{k,l} \leq w b_k \quad (8)$$

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

(3) 計算・ネットワーク資源の重みを最小化

(4),(5),(6)はX(計算機)の制約

(4) 選択サイトは1つのみ

(5) サイトは1回以上選択されない

(6) 要求CPU数が提供可能

(7),(8)はY(ネットワーク)の制約

(7) 要求パスがある場合  $y_{k,l}$  の総和は1以上, ない場合0

(8) 要求帯域が提供可能

(9)はX,Y両方の制約

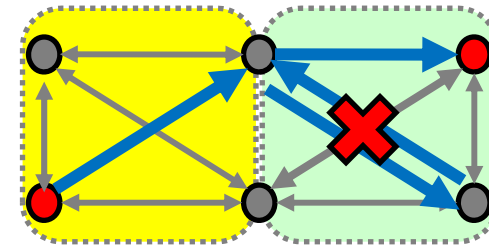
(9) 流量保存則から導く

# 追加の制約条件 (冗長パス探索の枝刈り)

Subject to

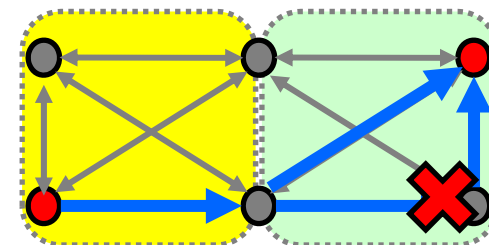
$$\forall l \in E_r, \forall m, n \in V (m \neq n), y_{(m,n),l} + y_{(n,m),l} \leq 1 \quad (10)$$

双方向の枝を同時に選択しない



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

枝の数の最大値  $P_{\max}$  を指定



$P_{\max} = 2$  のとき

# 発表概要

- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- 基本性能の評価
- まとめ

# コアロケーション対象

資源の種類	コアロケーション対象となる属性情報
計算機	プロセッサ/コア数, プロセッサアーキテクチャ, メモリサイズ, ディスクサイズ, OS, アプリケーションライブラリ, IPアドレス(レンジ)
ストレージ	スペースサイズ/I/Oスループット, スペース/ファイルアクセス許可/スループット/アクセスモード, 冗長性
ネットワーク	帯域, 遅延, スイッチング手法, 冗長性, VLANタグID

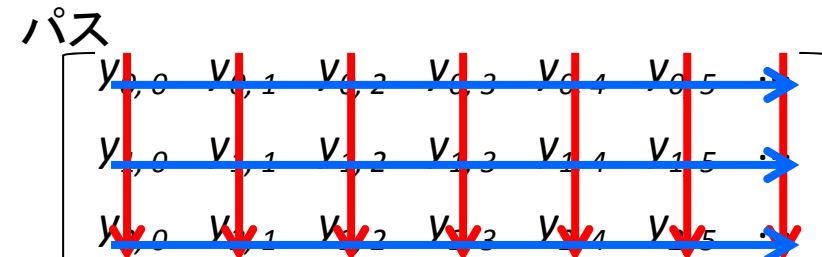
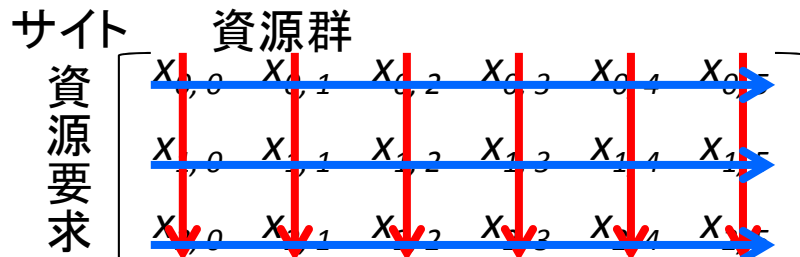
# コアロケーション手法の改良(1/2)

- 既存モデルに制約条件を追加することで、多様な資源・属性情報に対応
    - 各属性情報は全て計算サイト $x_{ij}$ またはパス $y_{ij}$ の資源パラメータと考える
      - ストレージに関するものは計算サイト $x_{ij}$ のパラメータ
    - 各属性情報の資源パラメータはスカラー値かブール値で表す

→式(12)-(19)
  - 目的関数は各コアロケーション対象の重み(E.g., 価格)を反映
- 式(20)

# 追加する制約式

資源の種類	制約式	スカラ/ブール	演算方向	コアロケーション対象
計算機	式(12)	スカラ	↓	プロセッサ/コア数, メモリ, ディスク, スペースサイズ/I/Oスループット, アクセススループット
	式(13)	スカラ	⇒	
	式(14)	ブール	↓	
	式(15)	ブール	⇒	プロセッサアーキテクチャ, OS, ライブラリ, 冗長性, アクセス許可, アクセスモード
ネットワーク	式(16)	スカラ	↓	帯域
	式(17)	スカラ	⇒	遅延
	式(18)	ブール	↓	
	式(19)	ブール	⇒	スイッチング手法, 冗長性

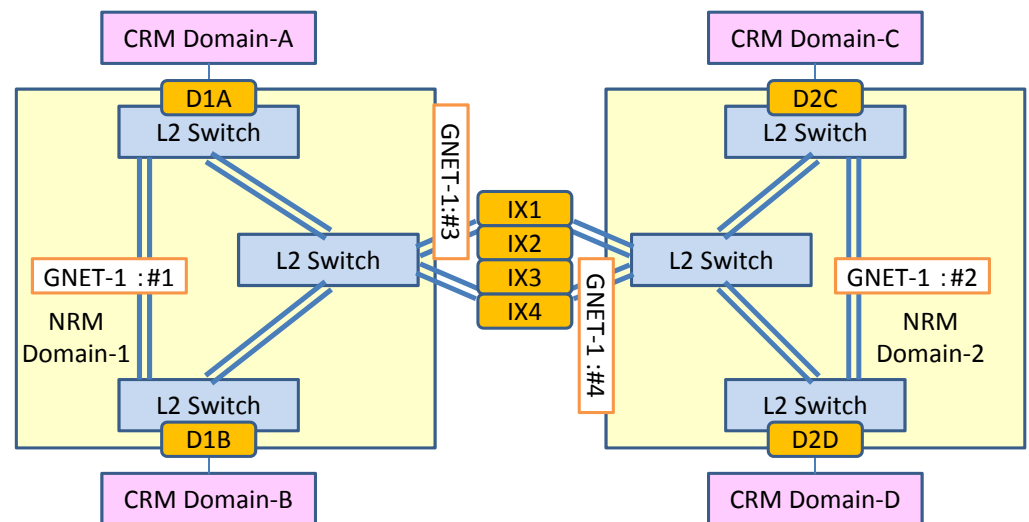


# コアロケーション手法の改良(2/2)

- ユーザの資源要求の一部指定への対応
  - j番目のサイトは"A"にしたい  
→  $x_{A,j} = 1, x_{i,j} = 0 (i \neq A)$
  - j番目のサイトは"A"か"B"か"C"にしたい  
→  $x_{i,j} = 0 (i \neq A, B, C)$

- 排他的資源への対応
  - 各拠点からIX1-IX4へのパスは1Gbpsずつ提供可能だが、いずれか1本しか通せない→

$$\forall ix \in IXs, \sum_{l \in Er, m \in V, m \neq ix} (y_{(m,ix),l} + y_{(ix,m),l}) \leq 2$$

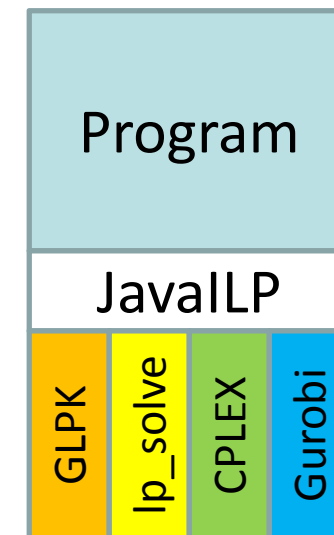


# 発表概要

- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- 基本性能の評価
- まとめ

# JavaLPを用いた提案手法の実装

- JavaLPを用いて複数のIPソルバを適用可能に
  - JavaLP (Integer Linear Programming)
  - 複数ソルバに対し, Javaの共通インタフェースを提供
    - GLPK, Ip\_solve, IBM ILOG CPLEX, Gurobi
  - ○: 単一コードで複数ソルバに対応
    - 入出力部分はJavaLPのライブラリを利用
  - ×: シンプルな操作のみ



# JavaLPを用いたプログラム例

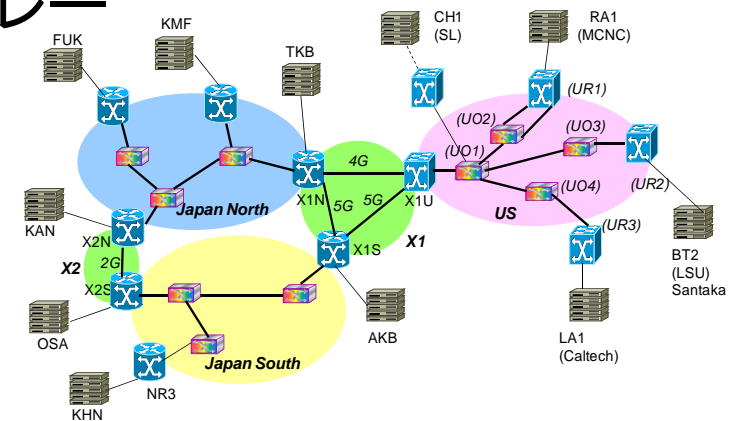
```
import net.sf.javailp.*;
public class JavaLPTest {
    public static void main() {
        Problem problem = new Problem(); // 問題の作成
        problem.setVarType(<変数名>, Integer.class); // 変数の設定
        :
        problem.add(<線形式>, <演算子>, <値>); // 制約式の設定
        :
        problem.setObjective(<線形式>, <Minimize/Maximize>); // 目的関数の設定
        SolverFactory factory = new SolverFactoryCPLEX(); // Solverの取得
        Solver solver = factory.get();
        Result result = solver.solve(problem); // 求解
        System.out.println(result);
    }
}
```

# 発表概要

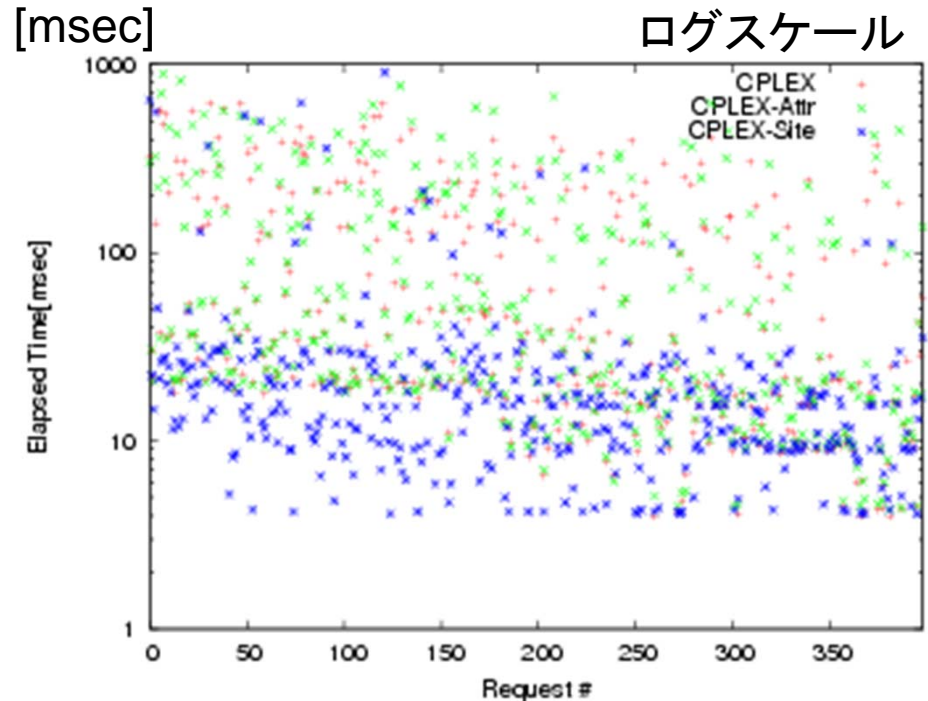
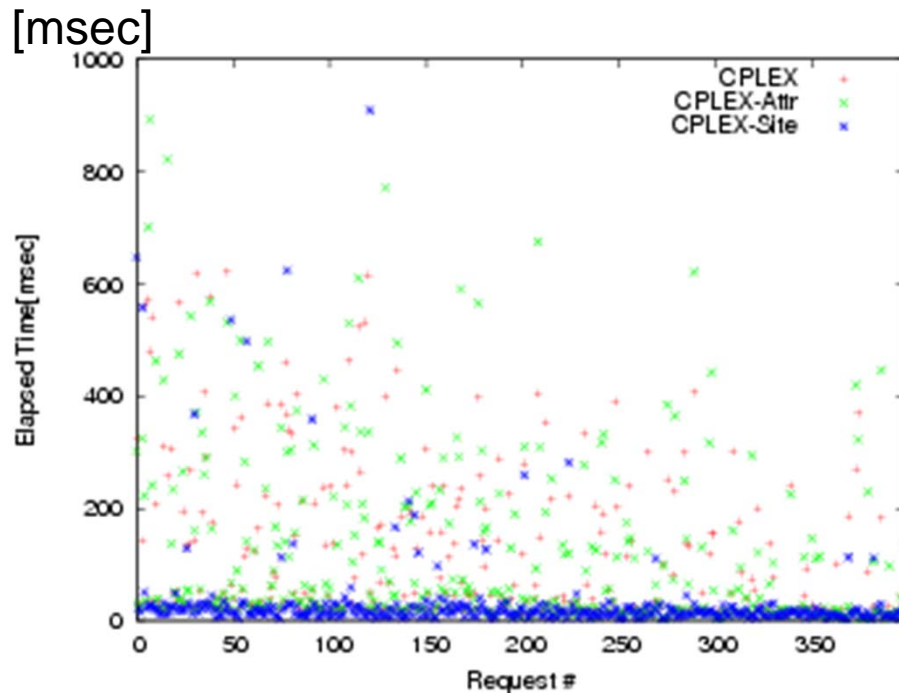
- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- **基本性能の評価**
- **まとめ**

# 基本性能の評価

- 提案手法におけるIPの求解時間を調査
  - 既発表手法の評価で用いたシミュレーションの1シーケンスを利用
    - 10計算サイト, 3ネットワークドメイン, 2ドメイン交換点
    - 2-4サイトの計算資源とその間のネットワークをランダムに要求
  - 指定なし/属性指定/資源の一部指定で比較
    - 属性指定ではプロセッサアーキテクチャを指定. 指定した属性を持つサイトは全体の3/4
    - 資源の一部指定では1つめのサイトをランダムに指定
  - 評価環境
    - Intel Core2 Quad CPU (2.66GHz), CentOS 5.3, 4GBメモリ
    - IBM ILOG CPLEX Optimization Studio V12.2, 32bit版



# 各資源要求における求解時間の比較



- CPLEX:指定なし, CPLEX-Attr:属性指定, CPLEX-Site:サイト指定
- いずれも1秒以下で求解
- 利用可能な資源が少なくなるにつれ, 求解時間短縮

# 求解時間の平均値, 最大値, 標準偏差

手法	平均値 [msec]	最大値 [msec]	標準偏差
CPLEX指定なし	96.8	914.6	238.7
CPLEX属性指定	106.2	860.0	269.5
CPLEXサイト指定	31.7	747.8	107.9

- サイト指定では, 平均値31.7msecと指定なしの3倍以上高速  
→現実的なシナリオ(E.g., データの所在でサイト指定)での求解時間短縮が期待
- 属性指定は指定なしと同程度

# 発表概要

- 前提とするコアロケーションシステムモデル
- 既発表コアロケーション手法
- 既発表コアロケーション手法の改良
- JavaILPを用いた提案手法の実装
- 基本性能の評価
- まとめ

# まとめ

- 多種資源とその属性情報を考慮したオンラインコアロケーション手法を提案
  - 0-1 IPにモデル化した手法を改良
  - ユーザの資源指定にも対応
  - JavaLPインタフェースを用い、複数のソルバを適用可能
- 基本性能の評価
  - 資源の一部が指定される場合は高速に処理可能
  - 属性指定がある場合も通常と同程度に処理
- 今後の課題
  - 大規模環境を想定した評価, 他のソルバとの比較
  - GridARS資源管理フレームワークへの実装

# 謝辞

- 本研究の一部は、科研費(21700047)および情報通信研究機構(NICT)の委託研究「ダイナミックネットワーク技術の研究開発」の助成を受けたものである