# JavaのためのP2Pライブラリ共有システム

高橋　右
林原研究室

# 背景と問題点

## ライブラリの導入

Jama

MeCab

OpenCV

Apache Commons

```
$ cd /usr/local/lib
$ sudo curl -O http://www.antlr.org/download/antlr-4.5-complete.jar
$ export CLASSPATH=“.:/usr/local/lib/antlr-4.5-complete.jar
$ cd
```

**ビルド時間やクラスパスの設定や管理のコストが大きい.**

# 背景と問題点

## PaaS

PaaS (Platform as a Service)

Edit，Code


ライブラリ，フレームワーク


**利点** ライブラリをコストなしで導入でき，協調開発できる

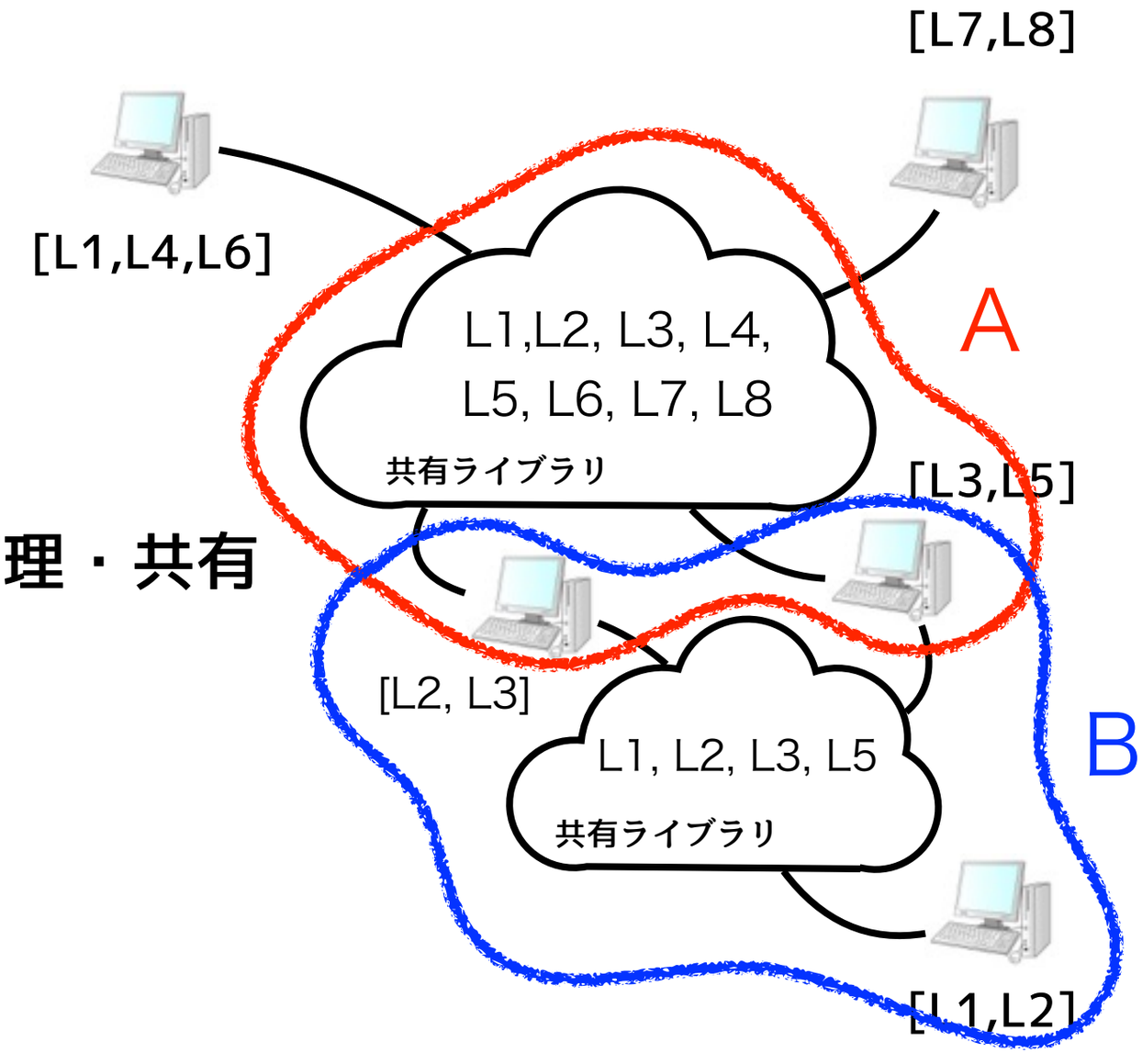
**欠点** コードを社外に出してしまうセキュリティ的リスク，ボトルネック

# 研究の目的

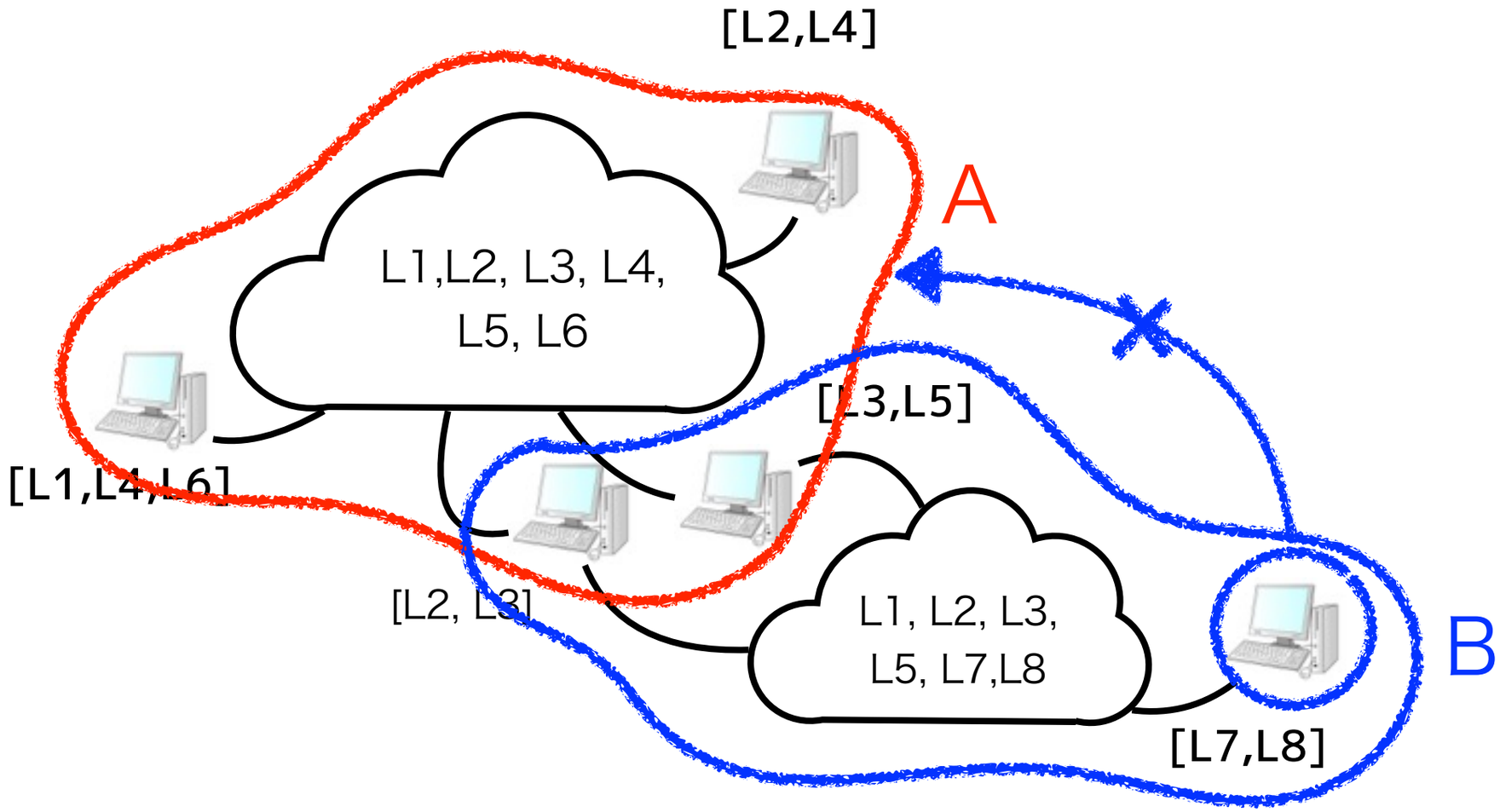
・P2Pシステム上でのライブラリの管理・共有

　・特定のグループごとにユーザ間で共有

・ライブラリの導入をサポート



# 概要

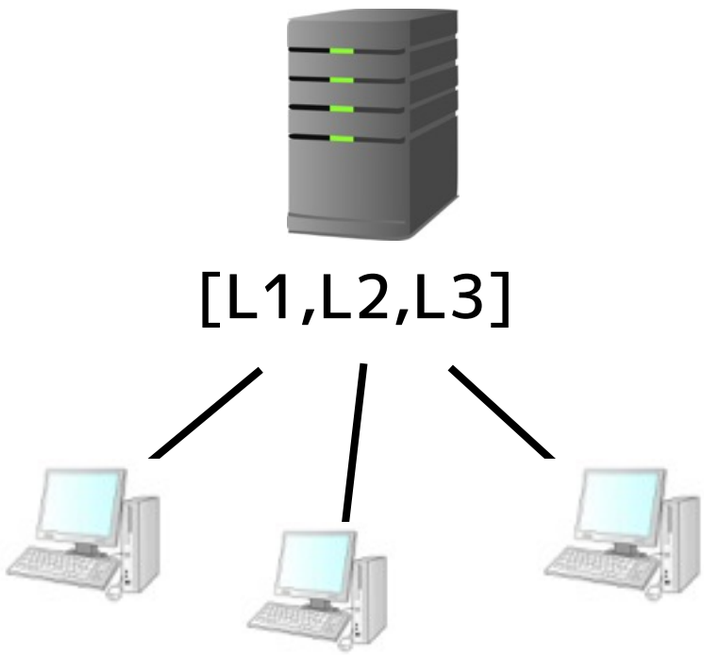



Javaライブラリを対象．グループ単位でのライブラリの管理・共有を行う．

# 提案システム

## 概要

PaaS

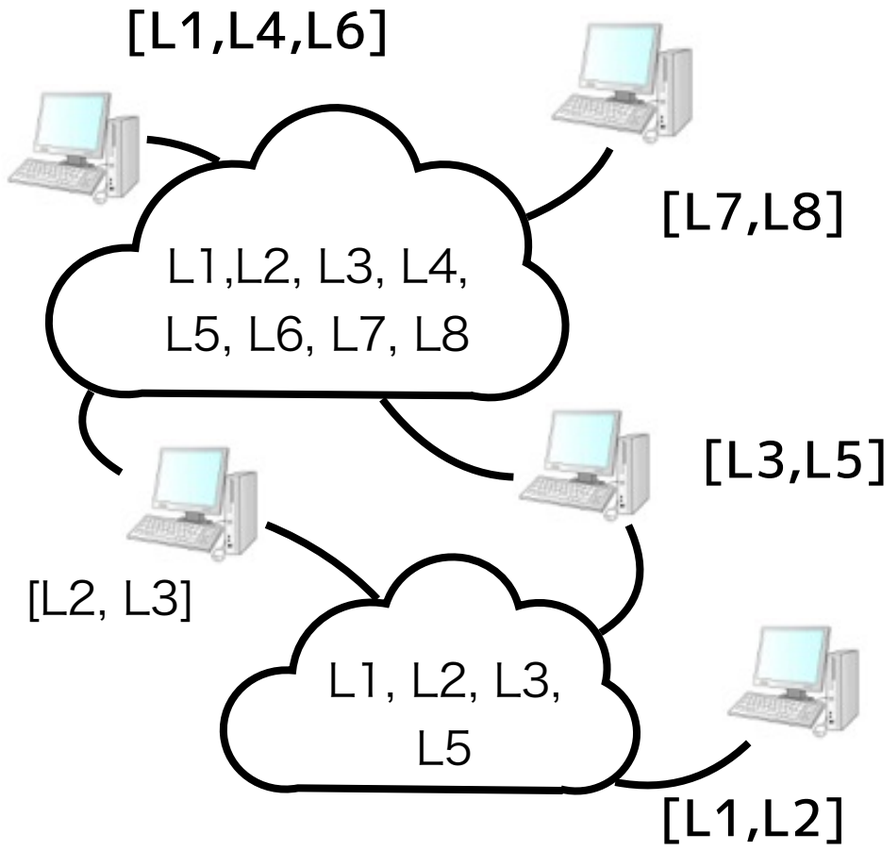



サーバ側で実行
（ソースコード転送）

単一障害点

ボトルネック

提案システム




ローカル環境で実行
（JavaRMI）

負荷分散

グループ管理

# ライブラリの管理・共有

## ライブラリの登録

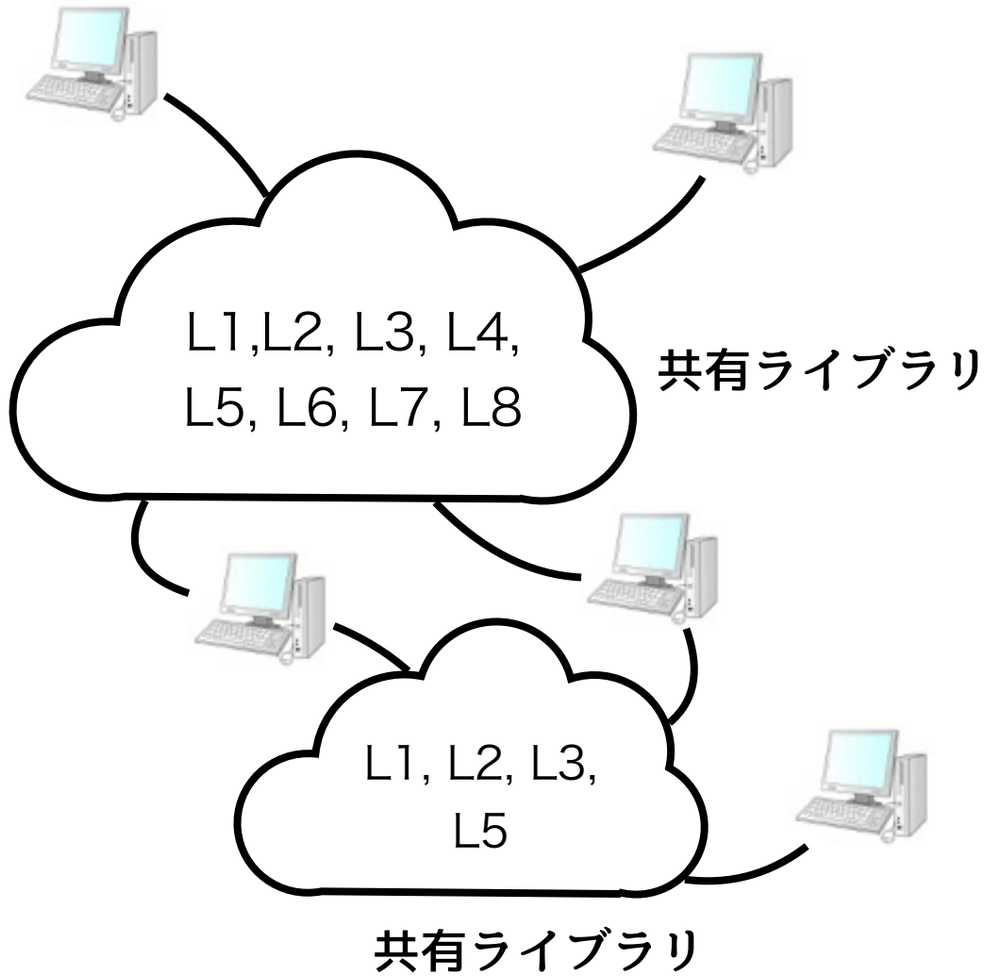
共有ライブラリに登録する

## バーチャル共有ライブラリ

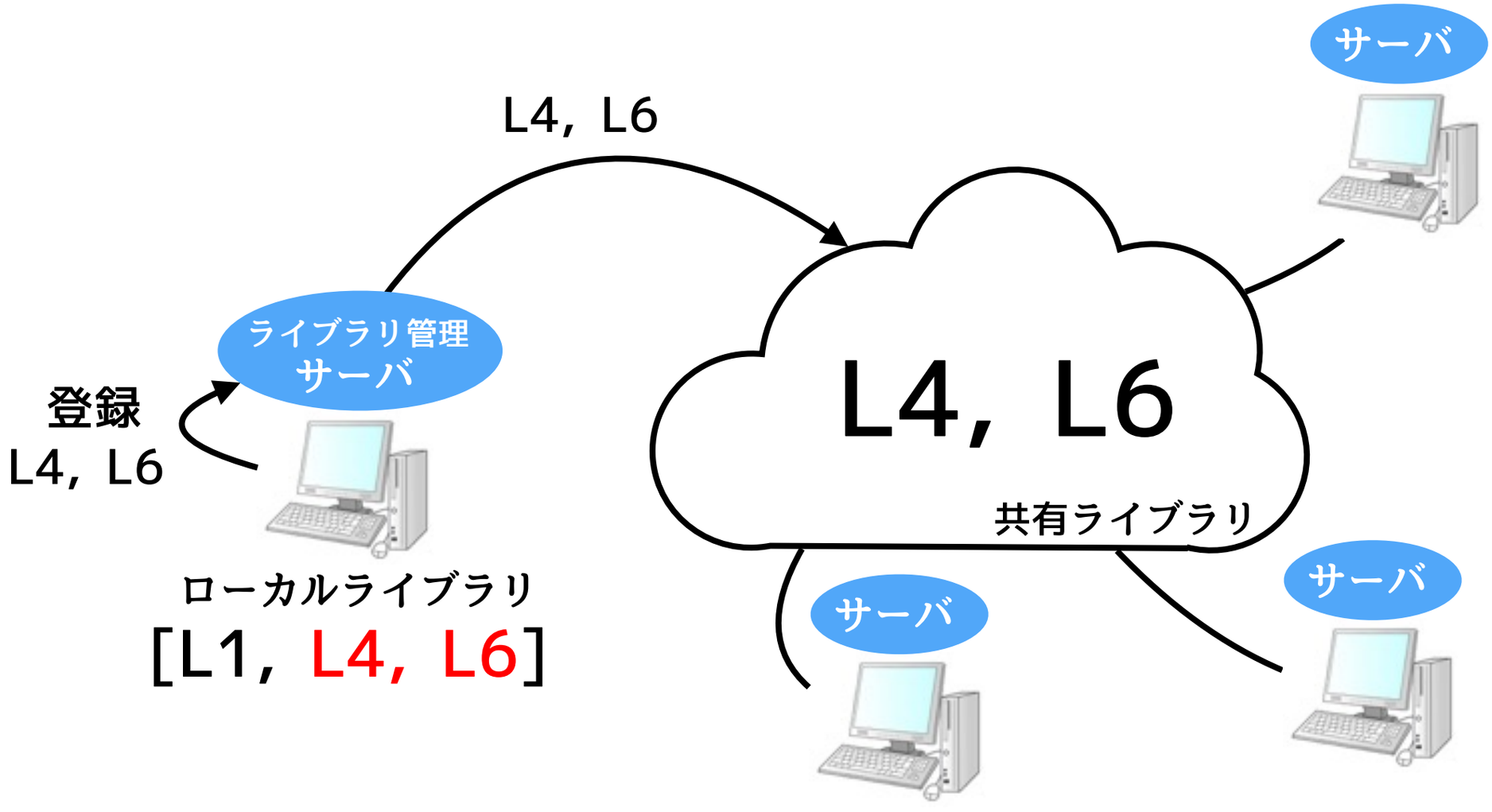
・リモート実行

・自動インストール

・手動インストール



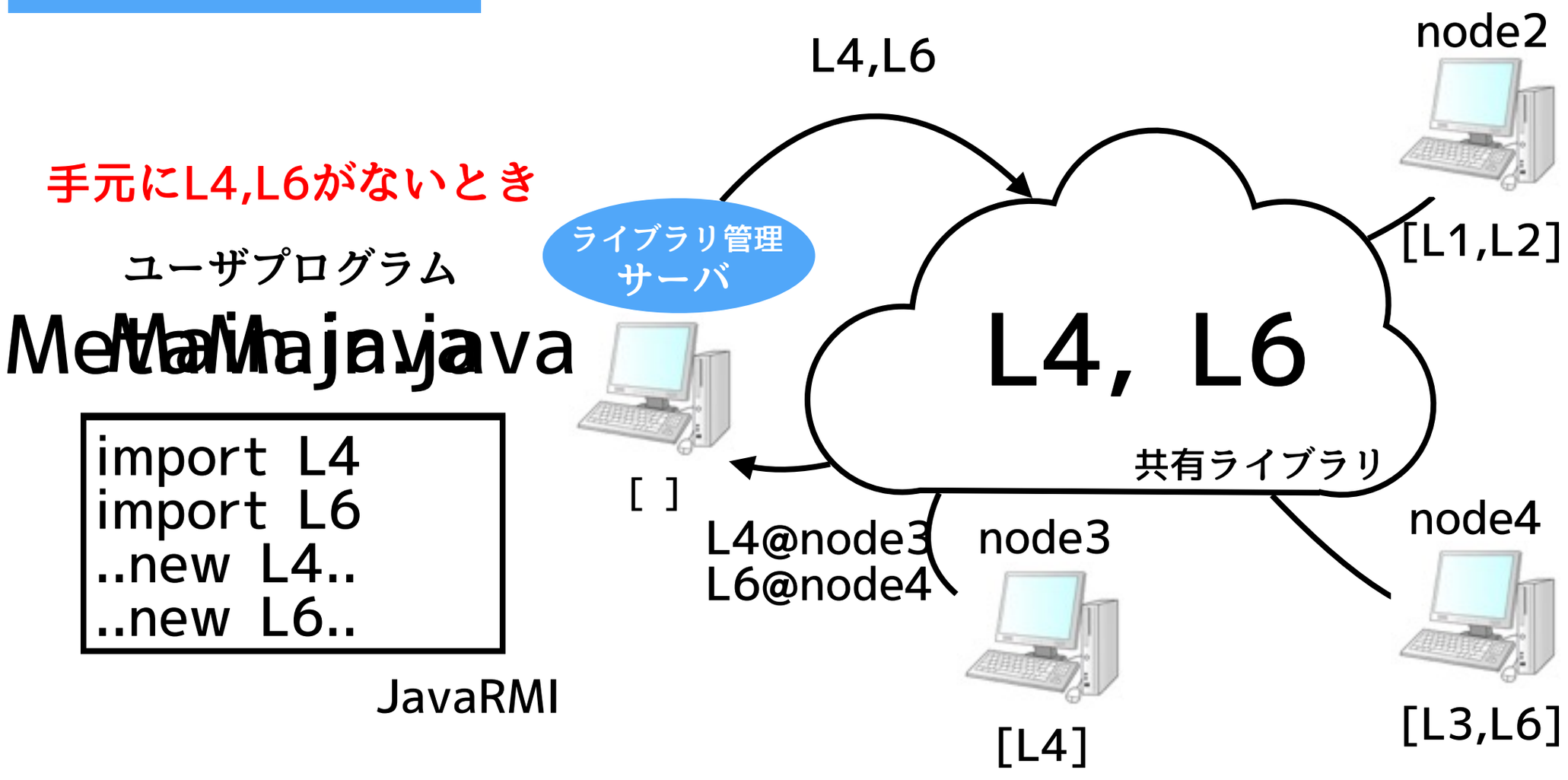
# ライブラリの登録




共有ライブラリへの登録によって，他の端末がライブラリを使用できる

# バーチャル共有ライブラリ

## リモート実行




ユーザのローカル端末にライブラリがなくても実行可能

-> 低速実行

# バーチャル共有ライブラリ

## リモート実行



Main.java

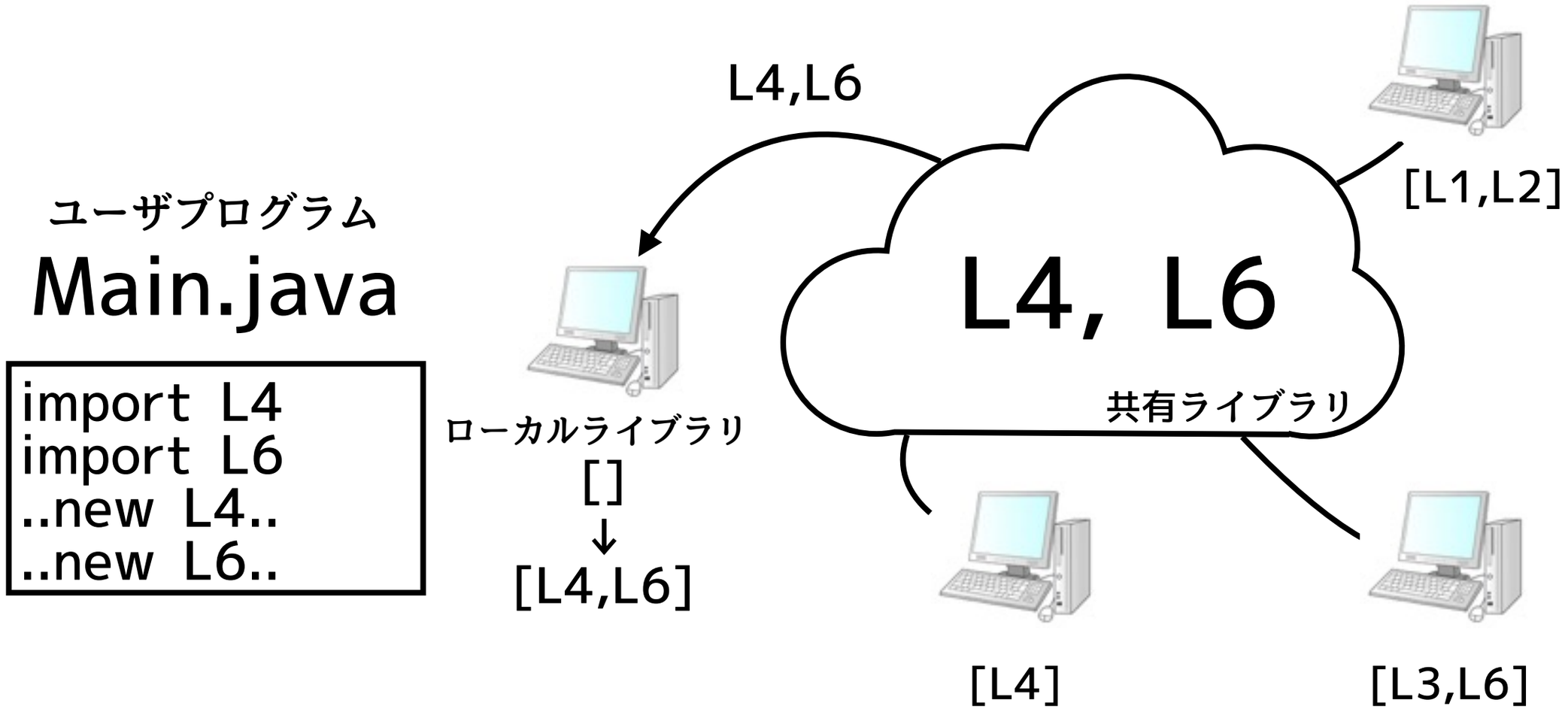

MetaMain.java

-> 独自の変換規則を定義

**リモート呼び出し用変換器によって，ユーザプログラムを通信用プログラムに変換する．**

# バーチャル共有ライブラリ

## 自動インストール

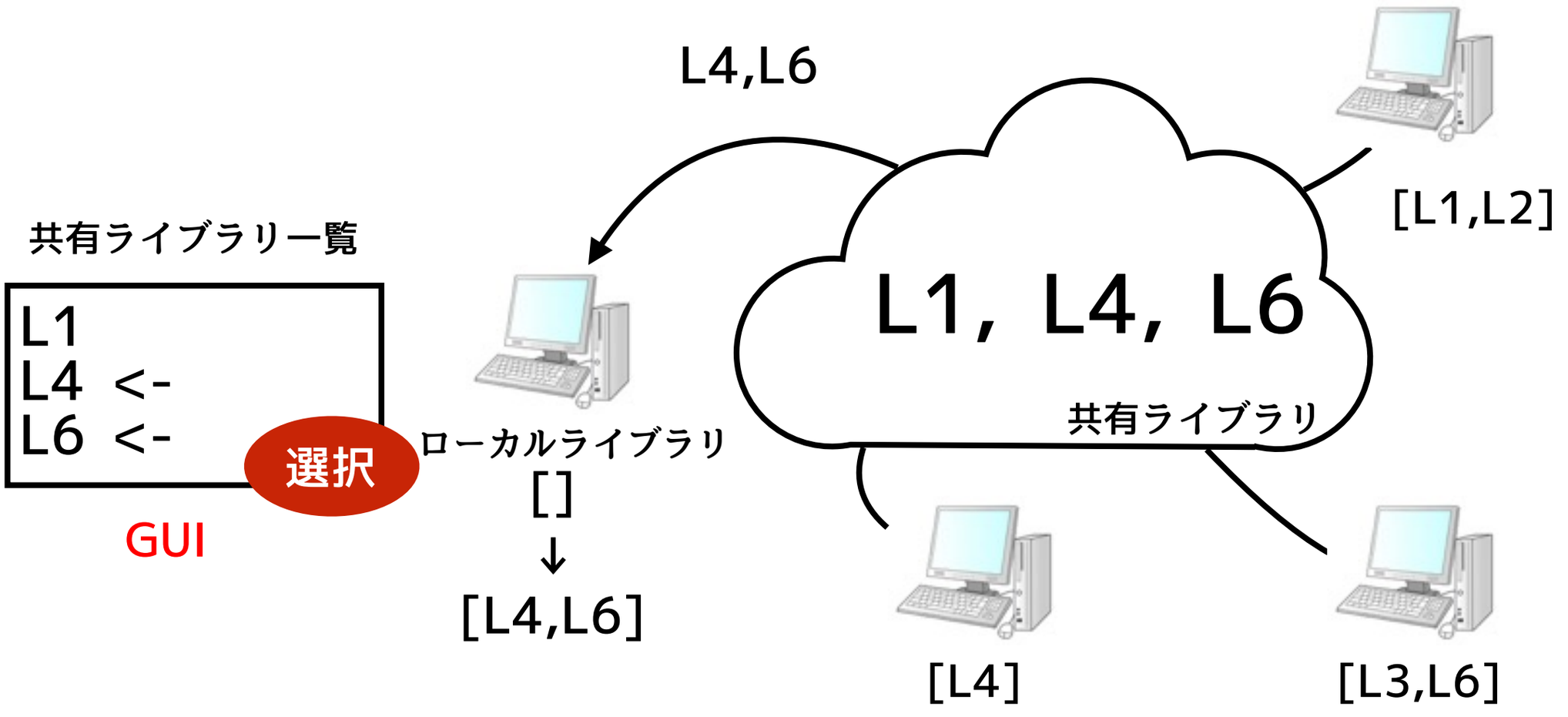



リモート実行の使用頻度の高いライブラリは自動インストールされる

-> 高速で実行（クラスパスの自動設定）

# バーチャル共有ライブラリ

## 手動インストール




ライブラリ選択し，手動での対話的インストールが可能

-> クラスパスの自動設定

# まとめと考察

**Javaのためのライブラリ共有システム**

**実装済み** リモート実行，自動，手動インストールのサポートの実装

**未実装** 通信を行うための自動コード変換器の実装，通信時の問い合わせの適切なルーティング，ライブラリ登録の自動化

# 参考文献

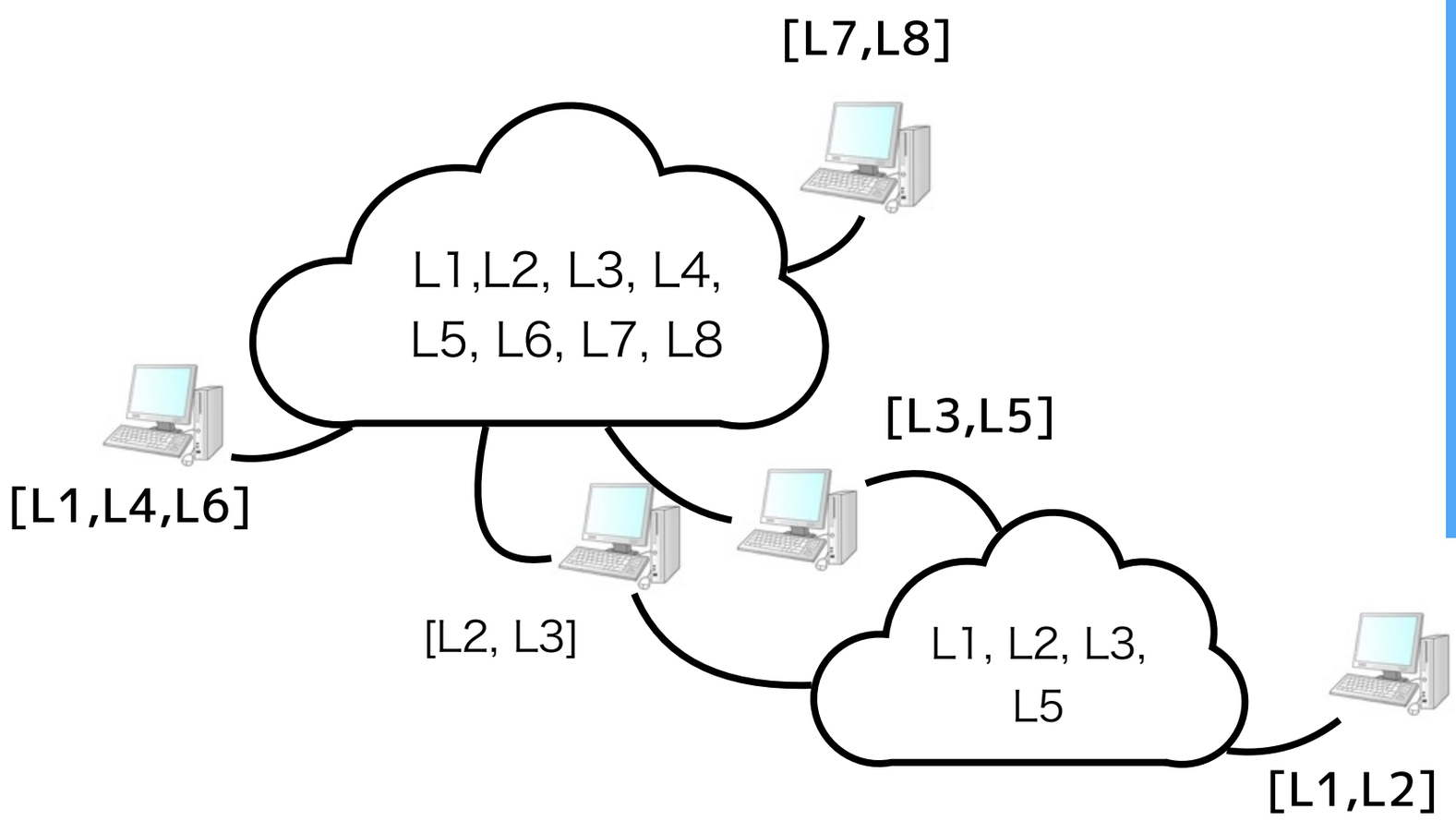
[1] 弘中 健,斎藤 秀雄,高橋 慧,田浦 健次朗:複雑なグリッド環境で柔軟なプログラミングを実現するフレームワーク,情報処理学会論文誌,p157-168 (2008).

[2] 高宮 安仁,松岡 聡:クラスタ設定のパッケージ化の設計と実装,情報処理学会研究報告. [ハイパフォーマンスコンピューティング] ,pp.55-60 (2004).

[3] ORACLE, Remote Method Invocation Home(online), available from ⟨http://www.oracle.com/technetwork/ java/javase/tech/index-jsp-136424.html⟩ (accessed 2014-11-14).

[4] PARR, T., ANTLR(online), available from ⟨http://www.antlr.org⟩ (accessed 2014- 11-14).

# 付録: ノード情報




**ノード情報**

- ライブラリ名s
- ホスト名，ポート番号
- ホスト名s，ポート番号s（隣接ノード）
- キャッシュ
- 距離
- スペック

# 付録: 変換規則

$$R1\begin{cases} IMPORT\,[qualified\,name] \\ \quad \rightarrow [None] \end{cases}$$

$$R2\begin{cases} CLASS\,[class\,name] \\ \quad \rightarrow CLASS\,[laas\,class\,name] \end{cases}$$

Example

$$R1\begin{cases} import\,Jama.Matrix \\ \quad \rightarrow [None] \end{cases}$$

$$R2\begin{cases} class\,Main \\ \quad \rightarrow class\,LaasMain \end{cases}$$

# 付録: 変換規則

$$R3 \left\{ \begin{array}{l} [method\ contents] \\ \rightarrow TRY\ \{[create\ remote\ object][method\ contents]\} \\ \quad CATCH\ \{[exception]\} \end{array} \right.$$

$$R3 \left\{ \begin{array}{l} [method\ contents] \\ \rightarrow try\{ \\ \quad Registry\ registry\ =\ LocateRegistry.getRegistry(host, port); \\ \quad LaasListener\ aLaasDB\ =\ (LaasListener)\ registry.lookup(\text{"}LaasDB\text{"}); \\ \quad [method\ contents] \\ \}CATCH\ \{\ [exception]\ \} \end{array} \right.$$

Example

# 付録: 変換規則

$$R4\begin{cases} [creator](args) \\ \quad \rightarrow [remote\,object].get[libname](args) \end{cases}$$

$$R5\begin{cases} [primary].[method](args) \\ \quad \rightarrow [primary].[reflection\,method](args) \end{cases}$$

Example

$$R4\begin{cases} new\,Matrix(array) \\ \quad \rightarrow aLaasDB.getMatrix(array) \end{cases}$$

$$R5\begin{cases} aMatrix.get(5,7) \\ \quad \rightarrow aMatrix.getMethod(\text{"}get\text{"}, newClass[]\{int.class, int.class\}).invoke(aMatrix, 5, 7) \end{cases}$$

# 付録: 関連研究

・複雑なグリッド環境で柔軟なプログラミングを実現するフレームワーク

・弘中ら，情報処理学会論文誌p157-168(2008)

・クラスタ設定のパッケージ化の設計と実装

・高宮ら，情報処理学会研究報告pp.55-60(2004)