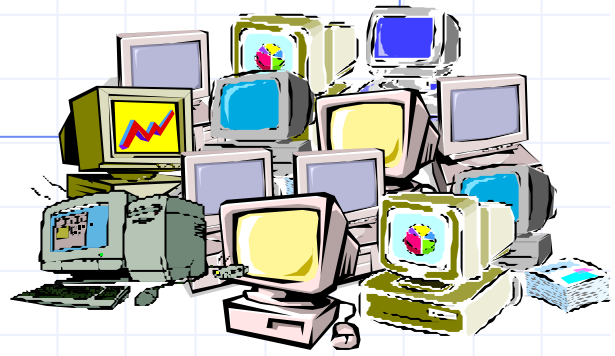
計算機資源の流通および集約のための P2P ミドルウェア

首藤一幸 大西丈治
田中良夫 関口智嗣

産業技術総合研究所 グリッド研究センター



Personal Power Plant

Grid Technology Research Center AIST

AIST

背景

- 計算機資源の流通に向けた動きが活発化
  - 資源: CPU能力、ストレージ、デバイス等
  - ASP (アプリケーションサービスプロバイダ) の出現
  - ユーティリティコンピューティングの提唱
    - 電気、水道のように、利用料に応じた課金
  - Globus Toolkitの認証、認可機能を利用した、PCクラスタ等の組織間での融通
    - スパコンのグリッドによる大規模な科学技術計算
      - 他組織へのジョブ投入や、MPICH-G2, Ninf-Gを用いた組織をまたいだ分散実行

AIST

動機

- PC をはじめとする個人情報機器では...
  - インターネット上分散処理プロジェクト
    - i.e. SETI@home, distributed.net, GIMPS, ...
    - 運営者だけがアプリケーションプログラムを用意できる。参加者は一方的に計算能力を提供する。
  - アカウントの発行
    - 計算機資源 提供側の手間が大きい。
    - 単一目的に機器群を連係させることは難しい。
      - i.e. 並列処理, センサーからのデータを別計算機で処理

AIST

計算機資源 授受のための ミドルウェア

- P3: Personal Power Plant
  - 個人情報機器の資源を容易に授受し、
  - 自由な並列プログラミングでそれらを連係、集約して活用するためのミドルウェア。
  - 資源提供者は、自らの意志で、どういった処理に資源を提供するのかを決めることができる。
    - 授受における重要な要件であり、P3 の特徴。
  - 応用例
    - 計算してくれる人、ストレージを貸してくれる人、カメラ等センサーを使わせてくれる人（資源提供者）の募集
    - インターネット上分散処理プロジェクトの運営
    - 組織内遊休 PC の cycle harvesting
  - 当面のターゲットは、PC 群を用いた並列・分散処理

AIST

オーバレイネットワークの利用

- 自由な並列プログラミングのために...
  - 昨今の IPv4 インターネットには、通信の相手や方向に制約がある。
    - Privateアドレス, DHCP: インターネット側から計算機を識別できない。NAT, NAPTゲートウェイがあっても、通信開始の方向は一方向。
    - ファイアウォール: IPv6 だろうと必要。
  - 通信には、P2P 通信ライブラリが提供するオーバレイネットワークを利用。
    - 下位通信網（TCP/IP, Bluetooth等）の差異、混在や、通信相手、方向の制約を吸収する。
    - 任意の計算機（ピア）間の双方向通信機能を提供する。
  - 本発表では、P2P通信ライブラリが提供する各種の概念を、本ミドルウェアの目的に対していかに適用するかを提案する。

AIST

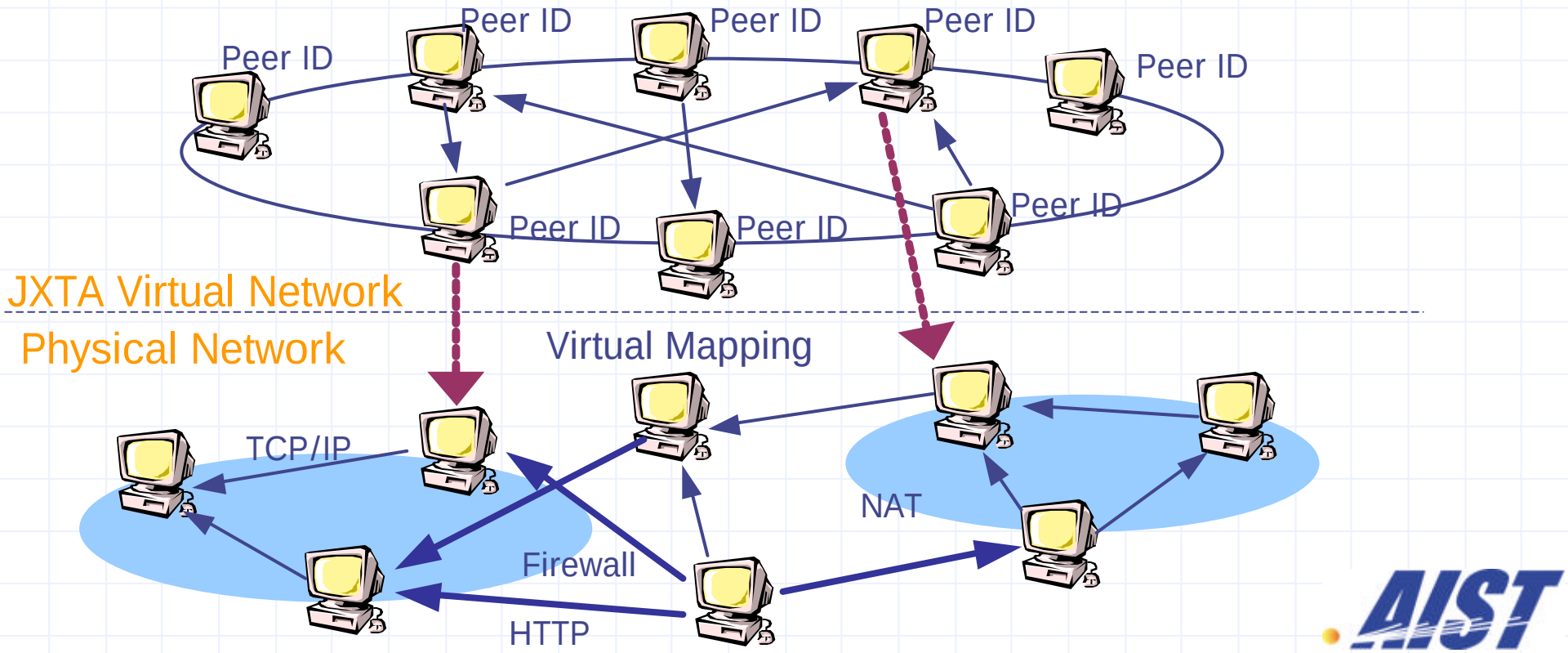
P2P 通信ライブラリ JXTA

- JXTA (じゃくすた) を採用
  - P2P ソフトウェアに共通の機能を提供するプロトコル、参照実装 (Java, C)、プロジェクト。
  - Sun Microsystems 社が 2001年に公開した。
  - (比較的) 安心して使える理由：
    - API が、2001年末の時点でほぼ固定されている。
    - 利用者が多い。
      - jxta.org でのユーザ登録は15,000 以上。
    - PCから携帯電話器まで、様々な規模の計算機で稼動する。
    - ライセンスが緩い：BSD由来である Apache と同じ内容。

JXTA の提供するオーバレイネットワーク

"Project JXTA 2.0 Super-Peer Virtual Network" 中の図

- 計算機はピアIDで識別される。
  - JXTAの実装は通信にTCPやHTTP, IPマルチキャストなどを用いるが、JXTAの利用者はそれを意識せずに、任意のピアを相手に通信を行う。
- ピアはピアグループを作成でき、複数のピアグループに参加できる。



JXTAの諸概念の適用法

- JXTAが提供する諸概念を、いかに本ミドルウェアの目的に適用するかを提案する。
  - ピアグループ、発見

- 「ピアグループ」
  - 並列処理には同報通信が欠かせない。
  - ジョブごとにピアグループを作成し、計算中のブロードキャストや、実行制御を、ピアグループ内同報通信によって行う。
    - cf. JNGI: 計算機を集合的に管理する目的にピアグループを使用。
  - また、このピアグループ内で、アプリケーションプログラムの配布を行う。
    - JXTA のファイル共有サービス CMS (Content Management Service) を使用。
    - そのピアグループが、ファイルの配布範囲となる。
      → ファイルを必要とする計算機と一致。

AIST

JXTAの諸概念の適用法

- 発見」
  - 資源提供者による、ジョブの発見
    - 計算機資源提供者は、資源の使用目的を決めることができるべき。
    - 資源利用者（ジョブ投入者）が作成、広報したジョブのピアグループを、資源提供者が発見する。
    - ジョブのピアグループを発見した資源提供者は、自らの意志に基づいて、そのピアグループに参加するか否かを決める。
  - 資源利用者（ジョブ投入者）による、提供されている計算機の発見
    - メッセージパッシング型アプリケーションの起動先指定に使用。
  - 資源提供者による、資源利用者（ジョブ投入者）の発見
    - 利用者のネットワーク上位置（IPアドレス等）を人手で事前に設定しておく必要がなくなった。

ミドルウェアの設計

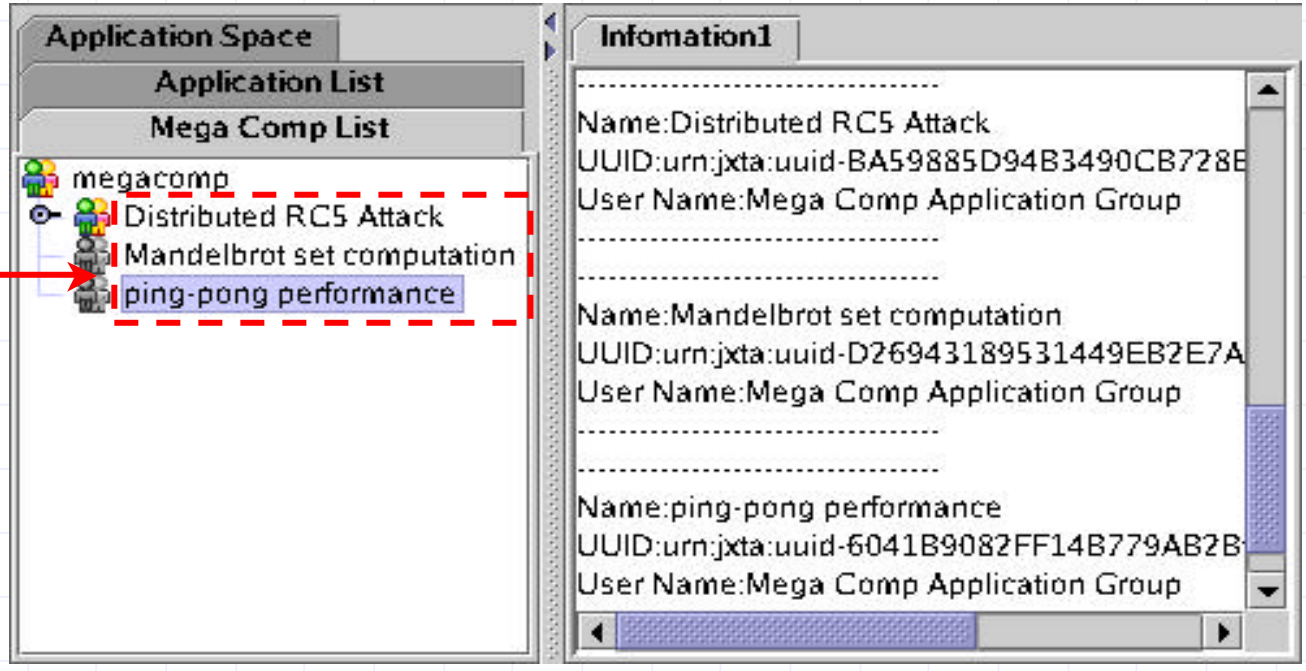
- ジョブ管理ソフトウェア
  - Host : デーモン
  - Controller : ジョブ投入ツール
- 並列プログラミング支援ライブラリ
  - マスタ･ワーカ
  - メッセージパッシング
  - これらのエミュレータ

AIST

ジョブ管理ソフトウェア：Host

- 資源提供者が起動しておくデーモン。
- ジョブのピアグループを発見し、提供者に提示する。
- 提供者は、シェルを用いて、ジョブを選択する。
  - [将来] ポリシー指定に従い、自動的にジョブを選択する。
    特に、集中管理が求められる組織内利用で重要。

投入されている
ジョブ

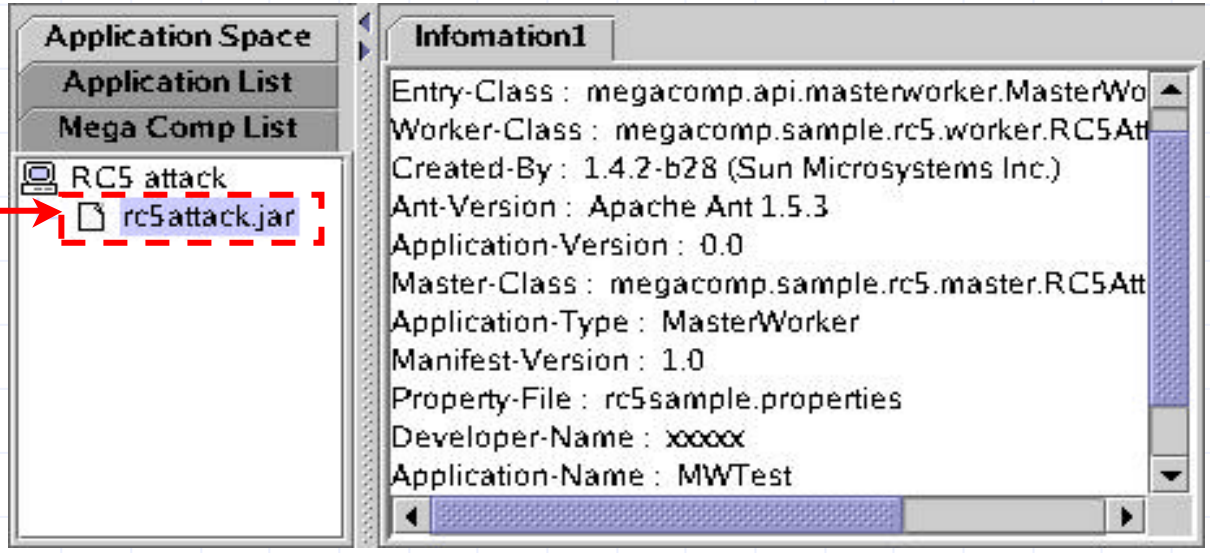


Host のシェル

ジョブ管理ソフトウェア：Controller

- 資源利用者がジョブ投入に用いるツール。
- アプリケーションプログラムを指定し、ジョブとして投入する。
  - アプリ：Java で記述し、JAR 形式のアーカイブに格納しておく。
    [将来] C, C++, FORTRANで記述、コンパイルしたネイティブコードのアプリも投入可能とする。

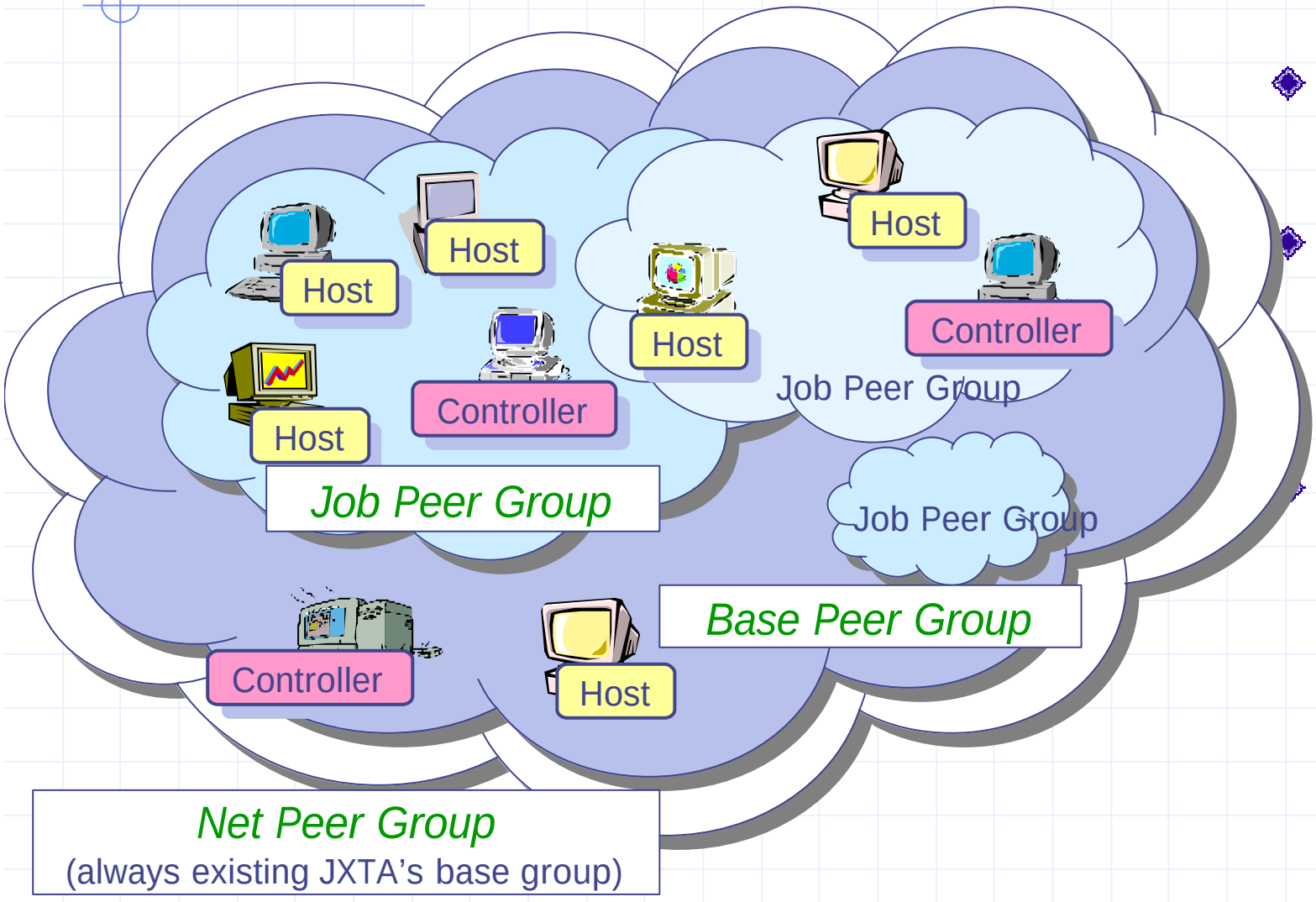
投入された
アプリケーション



Controller のシェル

AIST

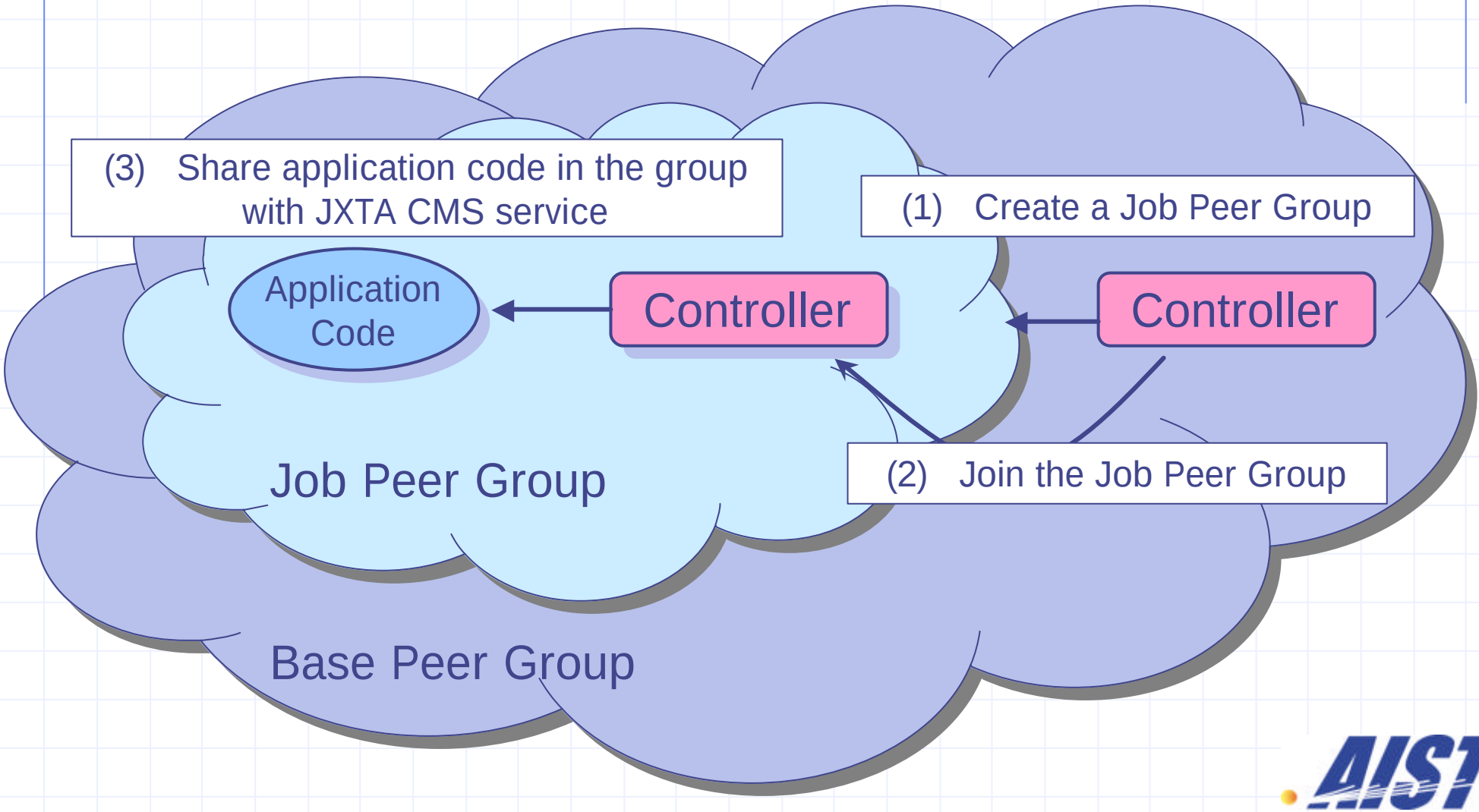
ピアグループの構成



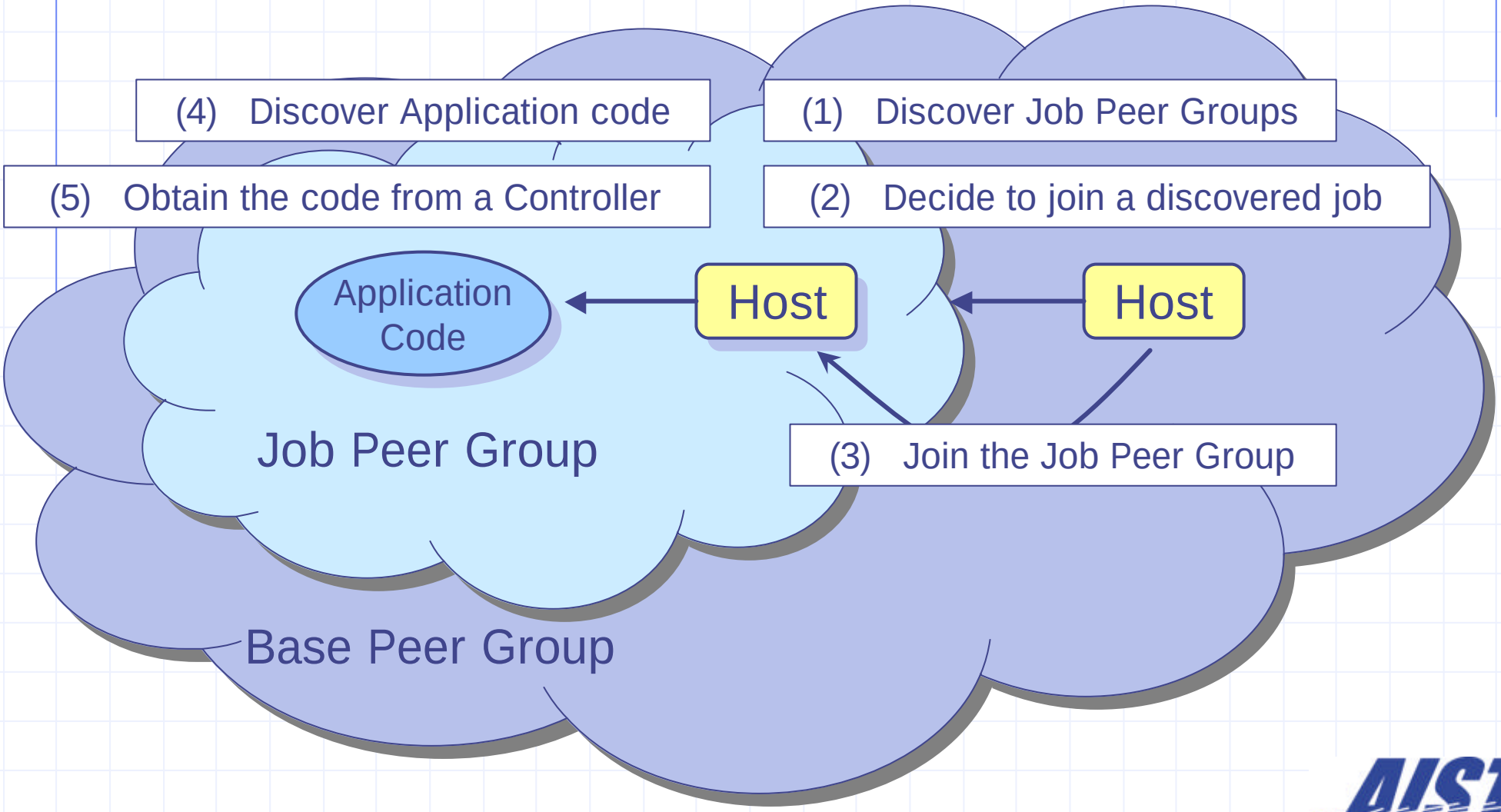
- Net Peer Group
  - JXTAで常に存在するグループ。
- Base Peer Group
  - P3 既定のグループ。全 Host, Controller は起動直後に参加する。
- Job Peer Group
  - ジョブの投入ごとに作成されるグループ。

AIST

Controller を用いたジョブの投入



Host の ジョブへの参加



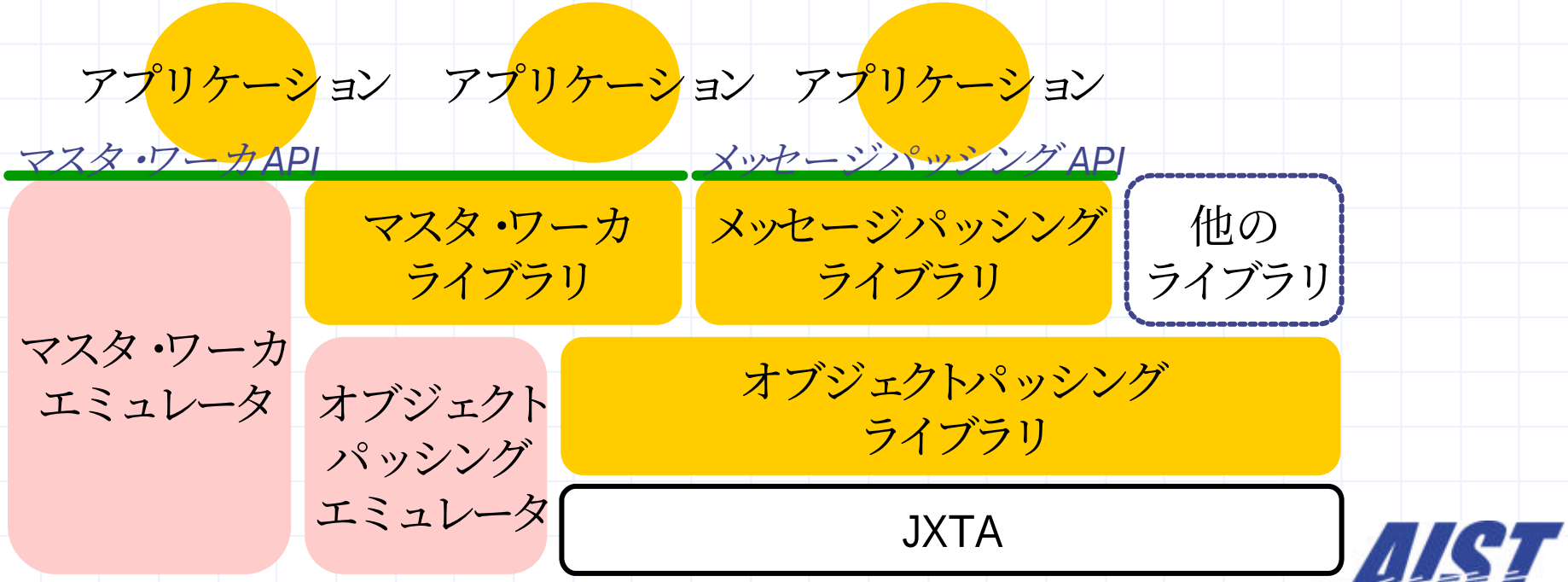
Controller を用いたジョブの起動

- 続いて、Controller のシェルから、ジョブの起動を指示する。
  - アプリが用いる支援ライブラリの種類によって、ジョブを起動できるタイミングが異なる：
  - ジョブ実行中の Host 参加を許す支援ライブラリ（i.e. マスタ・ワーカ）を用いるアプリの場合、Controller を用いたジョブ投入直後に起動できる。
    - Host の参加を待つ必要はない。
  - そうでない支援ライブラリ（i.e. メッセージパッシング）を用いるアプリの場合、ジョブ投入後、Host 群の参加を待ってから、起動する必要がある。

AIST

並列プログラミング 支援ライブラリ

- プログラマが直接利用するのは、マスタ・ワーカAPI と、メッセージパッシング API。
- エミュレータ
  - 並列プログラムを計算機 1台上で動作させる。
  - アプリの効率的な開発、デバッグに非常に有効。
    我々自身、マスタ・ワーカライブラリの開発に使った。そのまま実環境に載った。

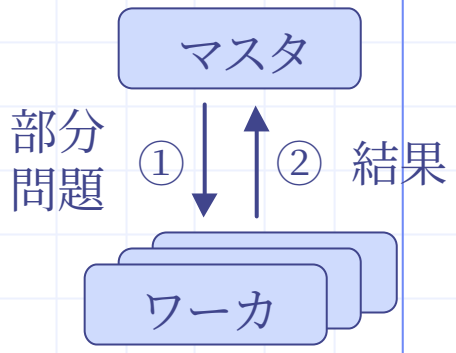


並列プログラミング 支援ライブラリ

- オブジェクトパッシング
  - （Java の）オブジェクトを送受信するライブラリ。
    - API: send(PeerID, Object), broadcast(Object), recv(PeerID), recv(), listenerの登録など
  - 通信相手は、JXTA のピアIDで識別する。
  - プログラマからは直接使わない。
  - この上に他のライブラリを開発する場合、JXTA の知識は不要。
- メッセージパッシング - JXTA-MPI
  - MPI にならった API。
  - 通信相手は、JXTA のピアIDの代わりに整数値 rank で識別する。
- マスタ･ワーカ
  - 後述

AIST

並列プログラミング 支援ライブラリ



マスタ・ワーカ

- マスタ・ワーカ型並列処理に特化した API とその実装（ライブラリ）。
- マスタは、部分問題（ワークユニット）を投入する。
  ワーカは、部分問題を処理し、結果を返戻する。
- 部分問題は要求に応じてワーカに配布する。
  - 動的負荷分散
  - アプリは部分問題のスケジュールを意識する必要なし。
- 障害と妨害への対応
  - 悪意あるワーカの存在を前提としている。
    - voting（投票）により虚偽の結果を検出できる。アプリ非依存。
  - タイムアウトに基づく、部分問題の再割り当てを行う。

AIST

マスタ•ワーカ API 使用例

- アプリのマスタ側

```
class Amaster implements Master {
  start() {
    while (…) {
      ProblemSet p = <a subproblem>;
      // submit a subproblem
      submitProblemSet(p);
    }
  }
}
```

- アプリのワーカ側

```
class Aworker implements Worker {
  ResultSet process(ProblemSet p) {
    // process a subpoblem and
    // return the result
  }
}
```

AIST

支援ライブラリの 性能評価

- [Q] オーバレイネットワークの機能はいいけど、通信その他が遅いんじゃないの？
- 測定の内容
  - 通信の遅延とスループット
  - マスタによる部分問題処理スループット
    - 単位時間あたりに、いくつの部分問題をさばけるか？
  - アプリケーションプログラムでの性能確認
    - 共通鍵暗号方式 RC5 の暗号化鍵総当り探索
    - 粒度の細かい部分問題への耐性を確認。
- 環境
  - 2.4 GHz Xeon × 34台の PC クラスタ, Gigabit Ethernet
  - Linux 2.4.19, Java 2 SDK 1.4.2, JXTA 2.1
  - 現インターネットよりリッチだが、それだけに、ソフトウェアによるオーバヘッドを明確に測定できる。

AIST

通信遅延

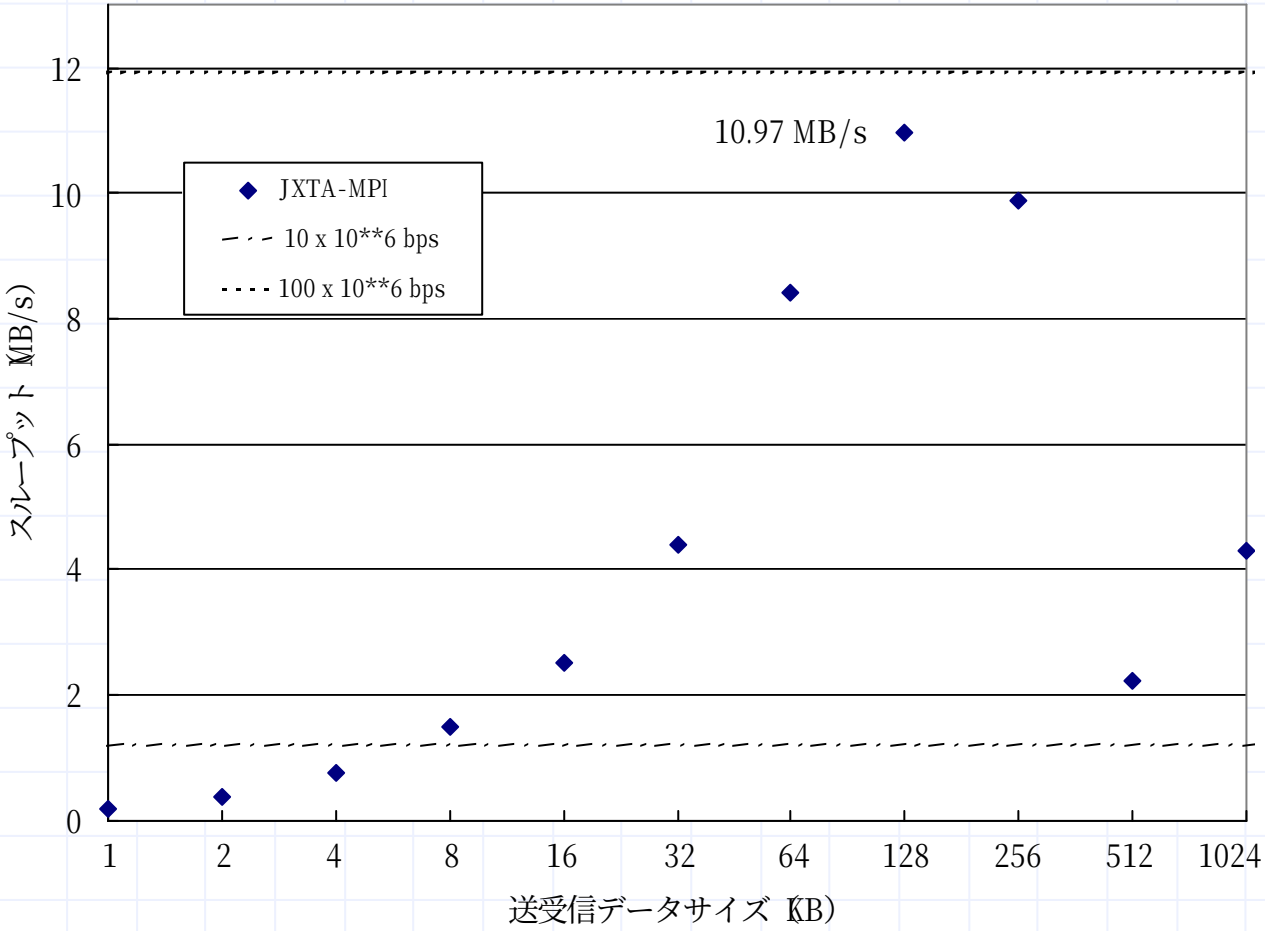
- 1バイトのピンポンを1000往復。片道あたり：
  - TCP (C プログラム)　　0.062 msec
  - TCP (Java プログラム)　　0.064 msec
  - メッセージパッシング　　4.5　　msec

- やはり、遅延は小さくない。
  - 送信：Java のオブジェクトをバイト列に変換し、JXTA メッセージに埋め込み、送信。
  - 受信：JXTAから呼び出された支援ライブラリ側のlistenerがJXTAメッセージを取得、バイト列を取り出し、オブジェクトを復元。支援ライブラリ中のキューに投入し、アプリに知らせる。
- この比較的大きな遅延が、マスタの部分問題処理スループット（後述）に大きな影響を与えている。

AIST

メッセージパッシングの スループット

- 100 × 10 ** 6 bps くらいは出ている。
  - 家庭、小規模オフィスのインターネット線や、低速LANを埋められるくらい。
- 送信データサイズが大きい場合に低下が見られる。
  JXTA の現実装の問題？
  - 分割して送ればよい。



[参考]
- TCP (C)
  108.0 MB/s
- TCP (Java)
  105.5 MB/s

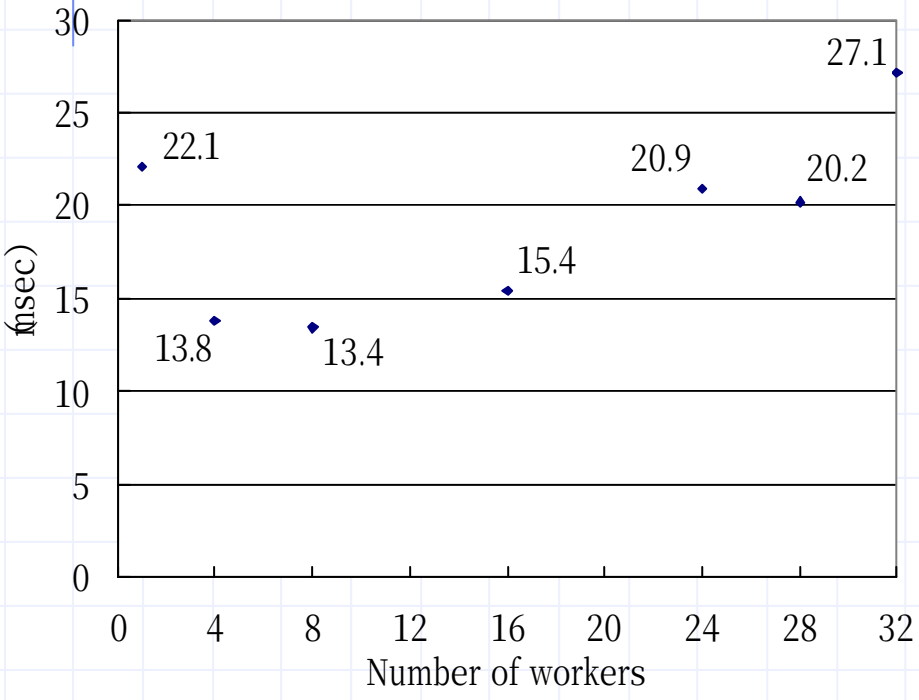
(送受信サイズ 1MB)

[注] K = 1024
M = 1024 ** 2

AIST

マスタの 部分問題処理スループット

- 並列処理効率のためには、マスタに集中する部分問題の配布、回収処理が律速とならないようにすることが重要。
- 単位時間あたりにさばける部分問題数が、マスタの性能を表す重要な指標となる。



- 実験：
  - アプリは、RC5 暗号化鍵総当たり探索。
    - 鍵空間を分割して部分問題とし、ワーカに配布。
  - 部分問題に含める鍵候補の数をゼロとした。
    - アプリ固有の処理にかかる時間を可能な限り小さくした。
  - 200 の部分問題を配布、回収し、1つあたりの時間を求めた。

AIST

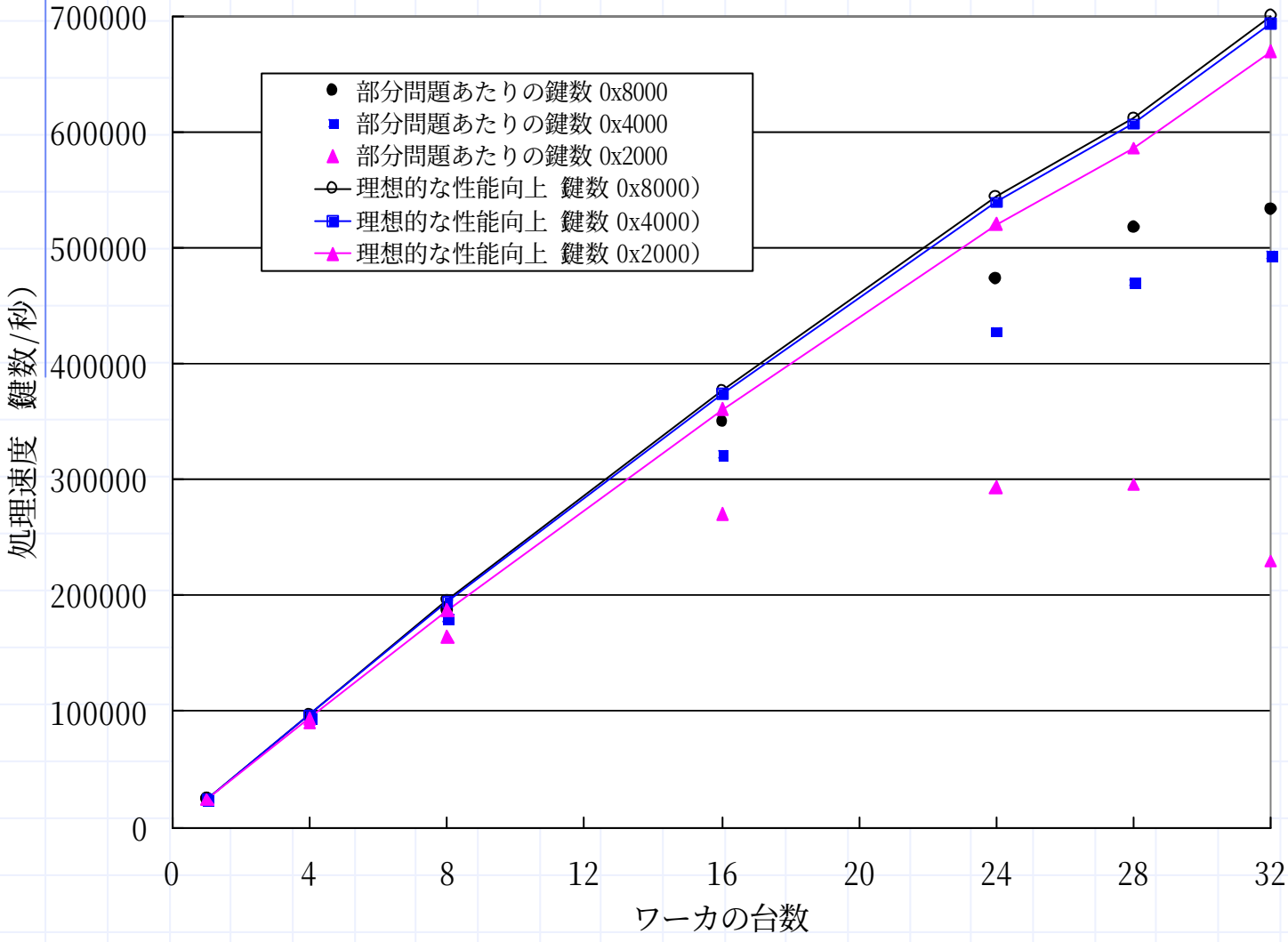
マスタの 部分問題処理スループット

- スループット
  - Best: 13 msec/個 77 個/秒
  - Worst: 27 msec/個 37 個/秒
  - 通信遅延だけで片道 4.5 msec かかっている。これがネック。
- United Devices 社のインターネット上分散処理プロジェクトの場合
  - UD社：抗ガン剤探索、インターネットサイトのストレステストのプロジェクトを運営。200万デバイスが登録されているとのこと。
  - 120 個/秒
  - Grid MP（ミドルウェア）に 4 CPU、DBに 4 CPU。
  - ミドルウェア自体の許容量は、最低でもこの数倍はあるだろう。
- 数万台規模のワーカを単一ジョブに参加させるなら、10 倍から100 倍の性能向上が望まれる。
  - また、複数のマスタでワーカ群を分担できるように、部分問題をマスタの外の DB に格納しやすいような設計をしている。

AIST

アプリケーションでの性能確認

- 暗号鍵探索アプリで、Hostの台数と、部分問題あたりの鍵数を変化させた。部分問題数は 200。
  - 部分問題の粒度を細かくしていった場合の、並列化効率の鈍り方を観察。



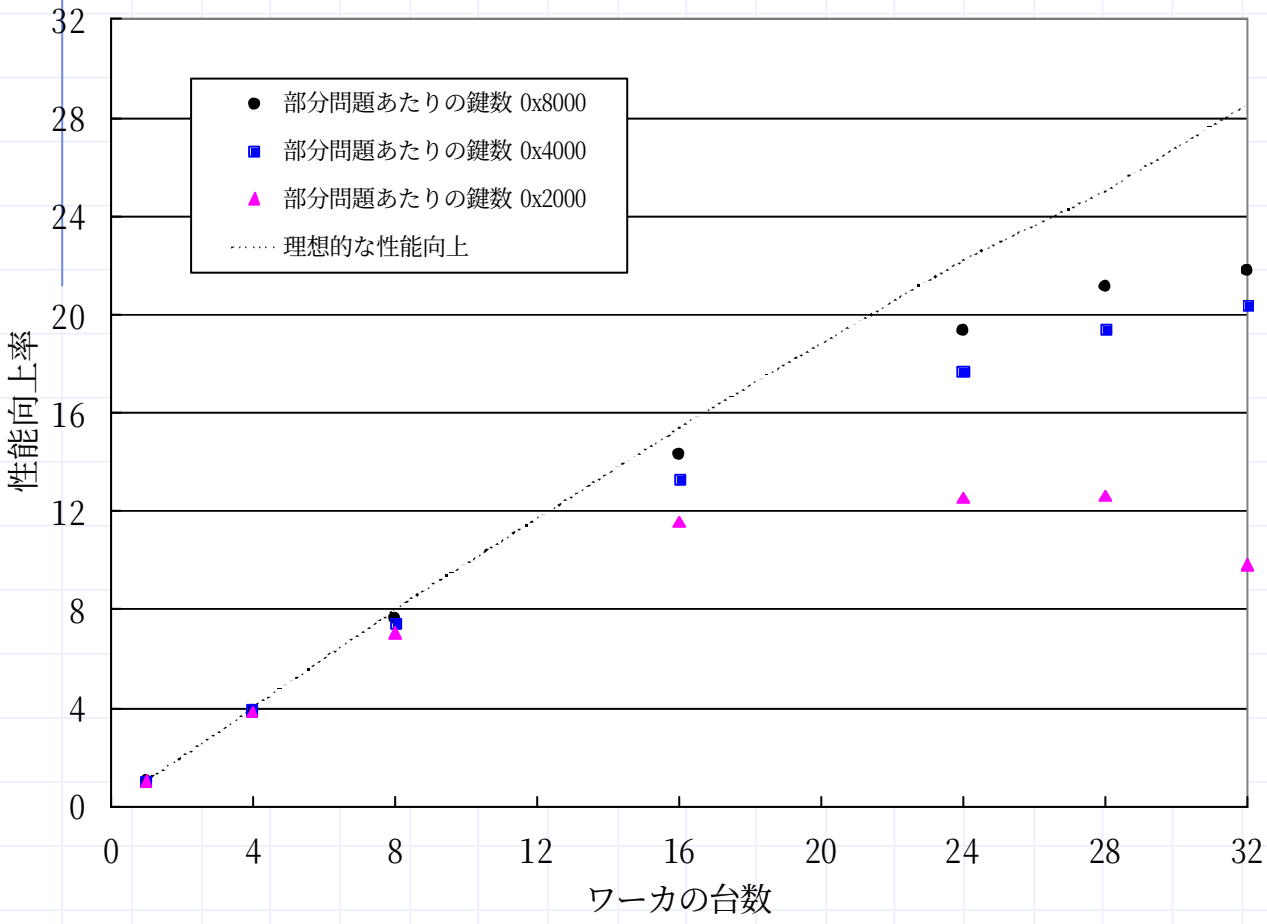
部分問題あたりの処理時間

- 鍵数 0x8000
  1.4 秒
- 鍵数 0x4000
  0.69 秒
- 鍵数 0x2000
  0.36 秒

AIST

アプリケーションでの性能確認

- Y軸を、性能向上率にしたグラフ：
  - 0.69 秒という比較的細粒度の部分問題でも、32台で20倍以上の性能向上は得られている。
    - cf. インターネット上分散処理では数時間～数日。
  - 粒度 0.36 秒では、24台で頭打ち。



部分問題あたりの処理時間

- 鍵数 0x8000
  1.4 秒
- 鍵数 0x4000
  0.69 秒
- 鍵数 0x2000
  0.36 秒

AIST

関連研究

- JNGI (by Sunの人)
  - JXTA を用いた分散処理フレームワーク
  - 計算機群の効率的な管理のためにピアグループを利用。
    - cf. P3 は、ジョブのピアグループ。
  - 2003年 7月現在、構想の主要部分がほとんど実装されていない。
    - ピアグループを活用しない。
    - 性能評価が信頼に足らない。Solaris, Windows, Linux を合わせて150台でリニアな性能向上？？？
- XtremWeb, GreenTea, Javelin, Bayanihan, Ninflet, ...
  - 資源提供者がジョブを選択できない。
  - プログラミングモデルは、マスタ・ワーカや分割統治に限定。
  - ファイアウォール越えが考慮されていない。
    - JavaRMI, HORB, TCP などを使用。
  - 悪意あるワーカ対策を打っていない or 不明。

AIST

まとめ

- 計算機資源を容易に授受し、自由な並列プログラミングでそれらを連係、集約して活用するためのミドルウェア P3 を開発している。
  - 資源提供者は、自らの意志で、どういった処理に資源を提供するのかを決めることができる。
- 自由なプログラミングのために P2P通信ライブラリ JXTA を採用し、P3 の目的に対してその諸概念をいかに適用するかを提案した。
  - ピアグループと発見

AIST

まとめ

- 懸念される性能を評価した。
  - 通信遅延は大きく、マスタの部分問題処理スループットを制限している。
  - 通信のスループットは、100 × 10 ** 6 bps 程度は出る。
  - マスタの部分問題処理スループットは数十/秒であり、改善が望まれる。
  - LAN では、部分問題の粒度が比較的細かい（0.7秒）場合でも、32台で20倍以上の性能向上を得られた。

AIST

今後

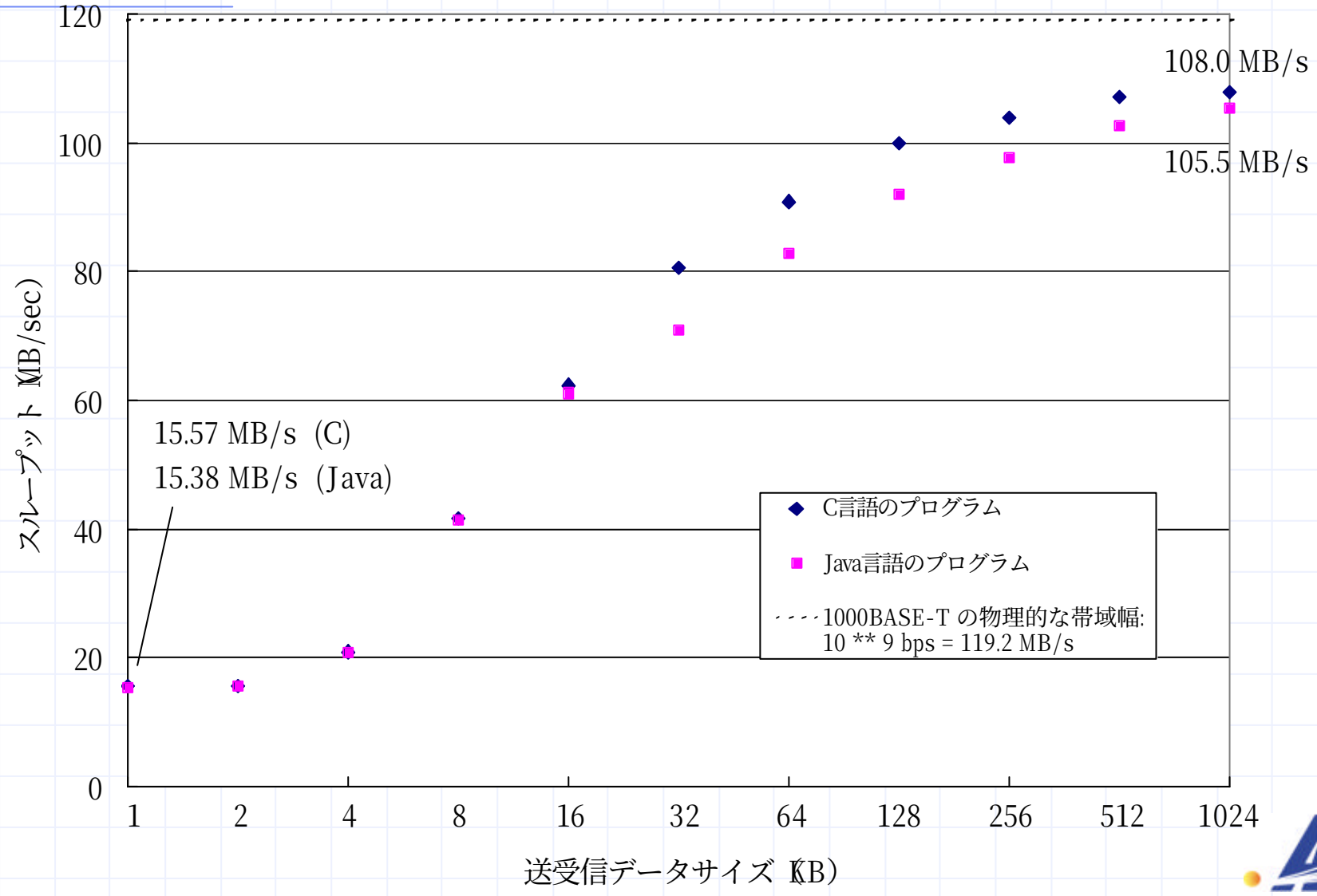
Personal
Power
Plant

- 研究的
  - 参加ジョブ選択ポリシー & Host の自動運転
  - ジョブのチェックポインティング
- 実用的な有用性、品質に向けて
  - 容易なインストールや、ミドルウェアの自動更新
  - ネイティブコードのアプリケーションへの対応
    - C, C++, FORTRAN アプリ
    - ネイティブコード用サンドボックスを使用して、資源提供者が安心してアプリを実行できるようにする。
  - アプリの実装を通じて要件を抽出し、P3 側からの支援を検討していく。
    - 例：気象のアンサンブル予報： 2XX MB の大きな固定データを使う。
- その他たくさん

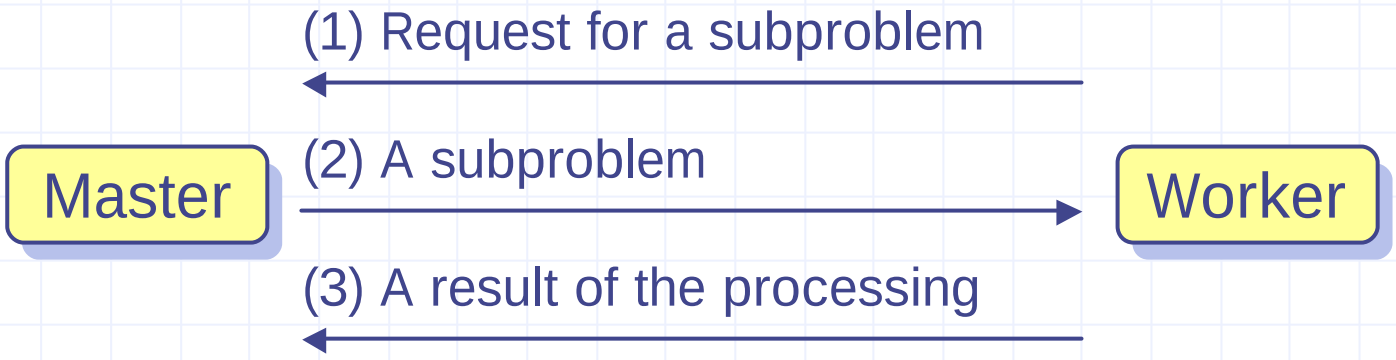
AIST

おまけ：TCP のスループット

- ピンポン1000往復。



おまけ：
マスタ•ワーカ間プロトコル



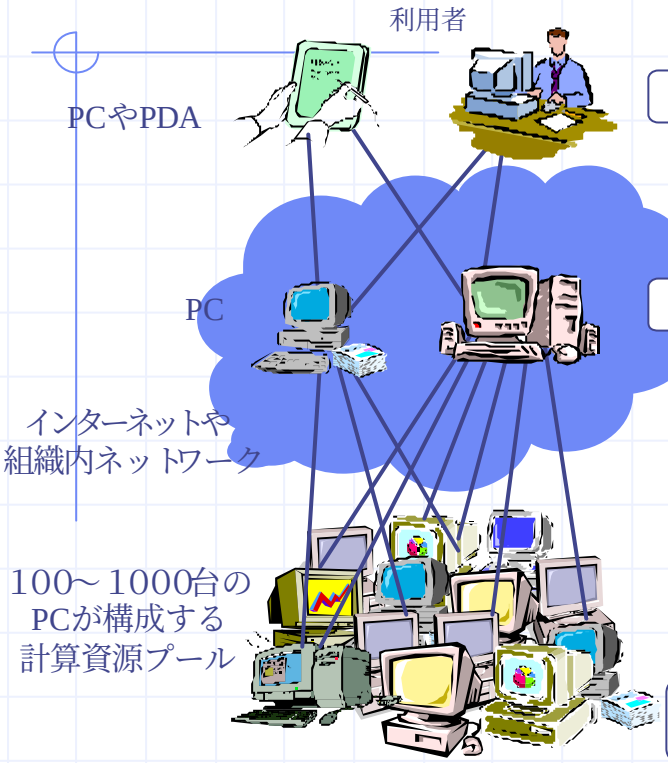
- 3-way
  - (3) と (1) をまとめて1メッセージで送ることで、2-way にはできる。
  - しかしそれだけでは、10 ～ 100 倍の部分問題処理スループットは得られない。

AIST

P2P Supercomputing

2003年 5月の
IPA Spring 用資料

稼動環境



ソフトウェアとその機能

アプリケーション制御ツール

•利用者はこれを用いて、ジョブの投入、削除などの管理作業を行う。

ブローカ

•アプリケーション制御ツールからのジョブ投入を受け付ける。
•実際の計算を計算サーバに依頼する。

計算サーバ

•各PCで稼動し、管理サーバから依頼された計算を行う。

プログラミング支援ライブラリ

•応用プログラムをサポートし、通信など応用の本質ではない処理を行う。

✓ 概要

遊休PCを用いた自由で容易な大規模分散処理を支援するミドルウェア。多量の計算やストレージを必要とする問題を抱えた利用者が、身のまわりやインターネット上のありあわせのPCを多数利用して、自分自身の手で、自由なプログラミングで分散処理を行うことを可能とする。

✓ テーマ名　組織内およびインターネット上の遊休PCを用いた大規模並列計算のためのミドルウェア

✓ 開発者　　独）産業技術総合研究所

AIST