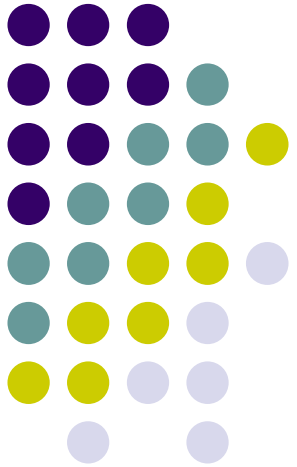
グリッド上における仮想計算機を用いたジョブ実行環境構築システムの高速化

山形育平†
高宮安仁†
中田秀基††,†
松岡聡†,†††

†: 東京工業大学
††: 産業技術総合研究所
†††: 国立情報学研究所

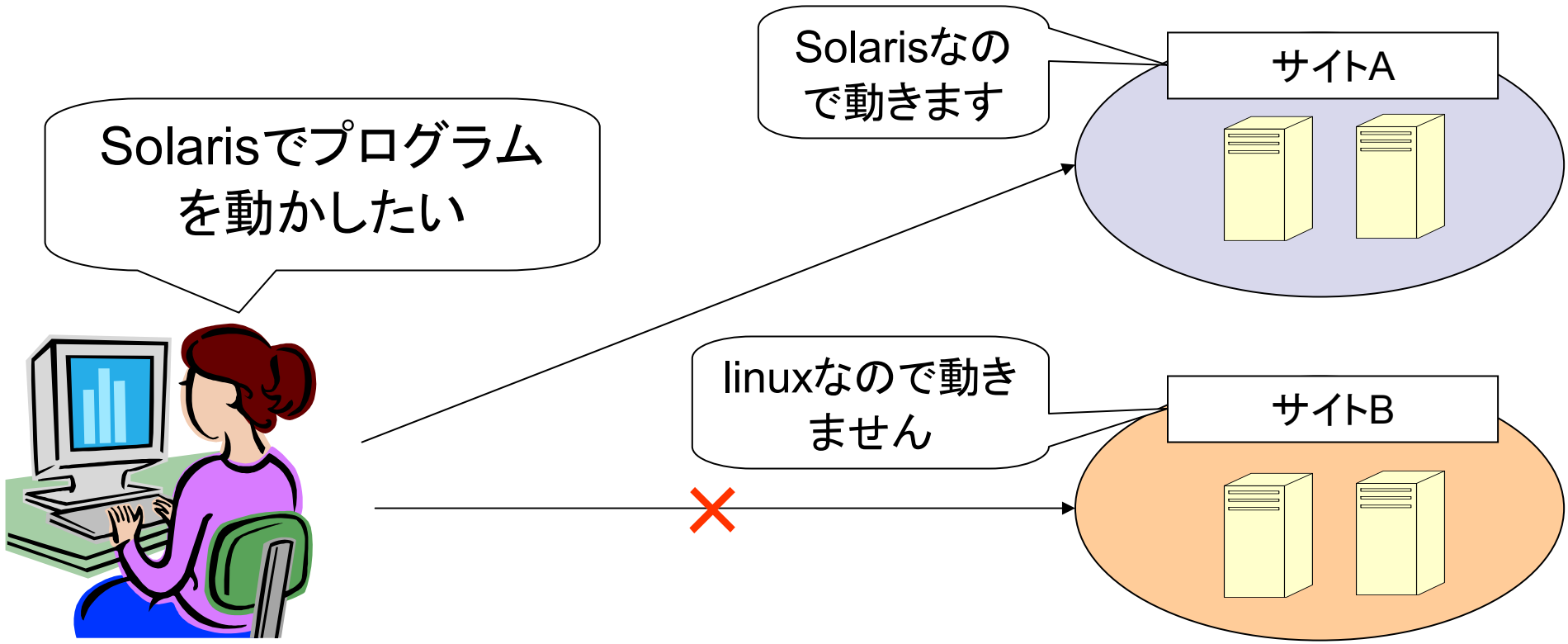


背景

- グリッド技術の普及
  - 複数ユーザがネットワーク接続された計算機資源を共有する機会が増加
  - ユーザが利用するOSやライブラリが多様化
    - 各計算機にインストールされている必要がある
  - 各計算機間で管理ポリシーが異なる
    - インストールされているソフトウェアは管理ポリシーに依存

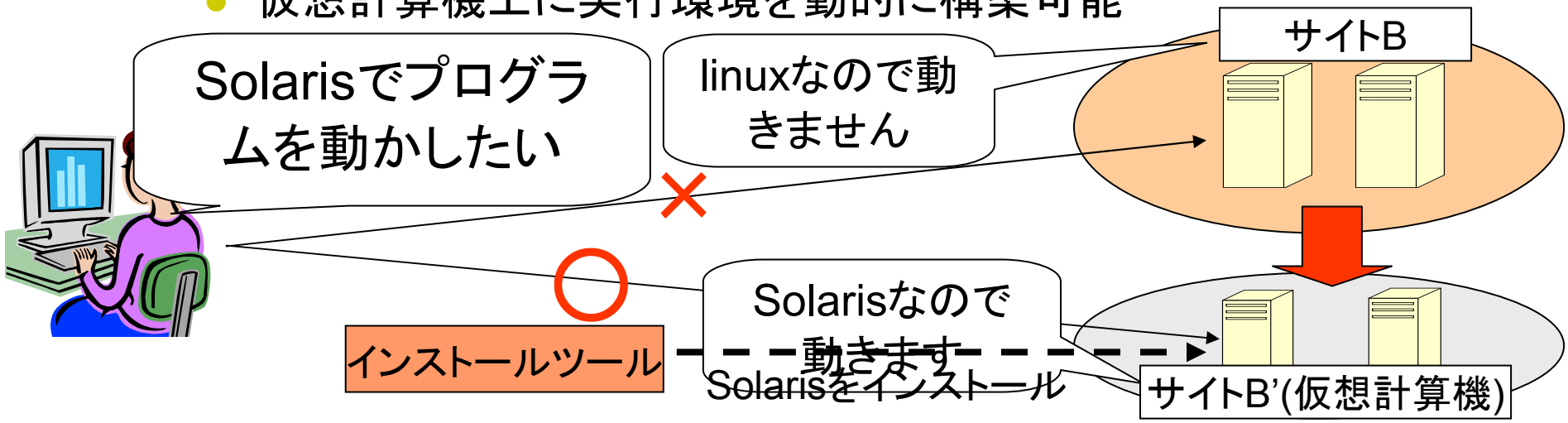
既存グリッドの問題点

- ユーザの求めるジョブ実行環境が常に提供されているとは限らない
  - 各管理サイトのポリシーによる提供されている環境の相違

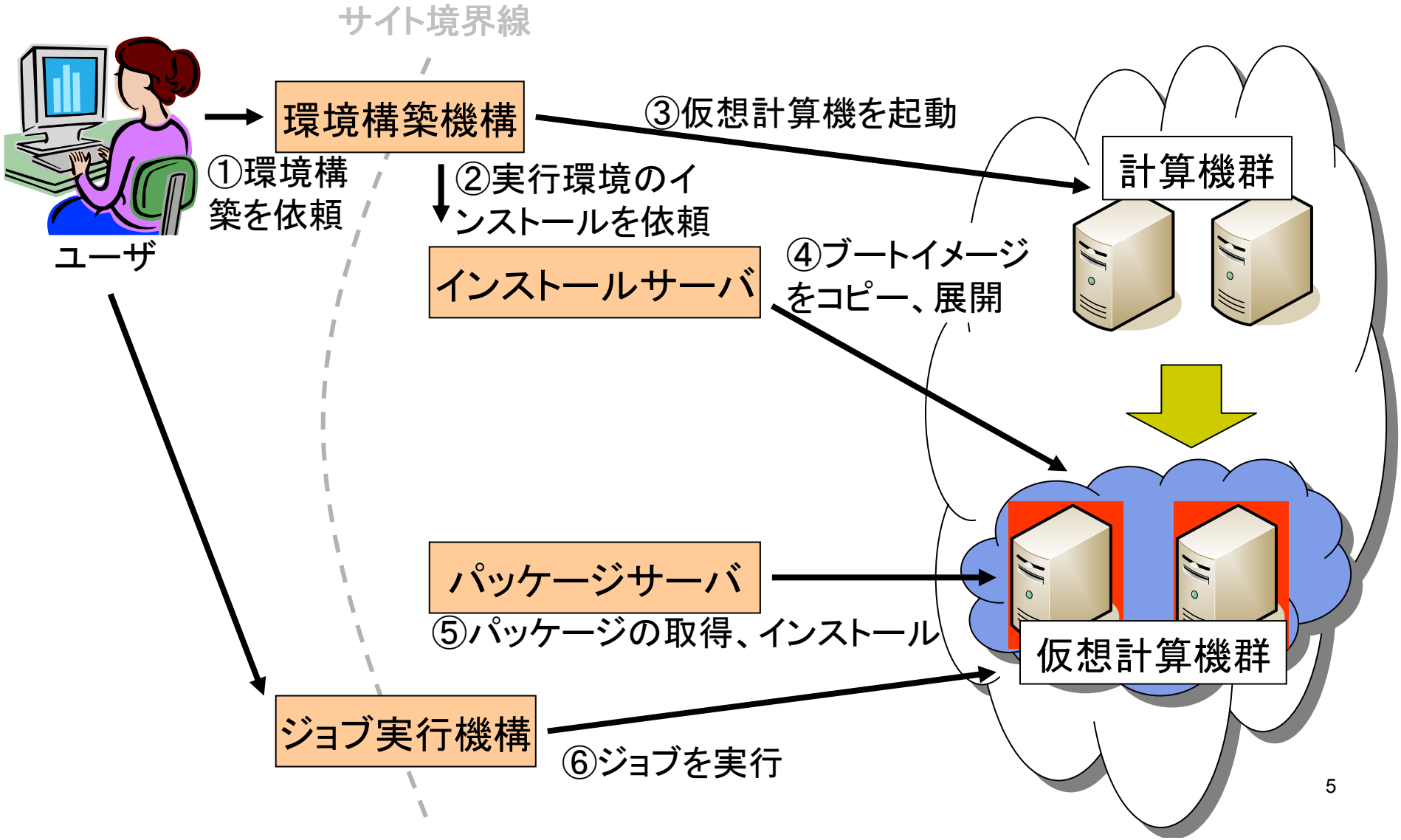


ORE(Open Resource Environment) Grid[SWoPP '05]

- 仮想計算機システムを用いてグリッド上にジョブ実行環境を動的に構築
  - 仮想計算機システムとは物理的な計算機の上に仮想的な計算機を構築するシステム
    - 既存環境を破壊しない
    - サイトポリシーに影響を受けない
  - 環境構築に自動設定・インストールツールを使用
    - 仮想計算機上に実行環境を動的に構築可能



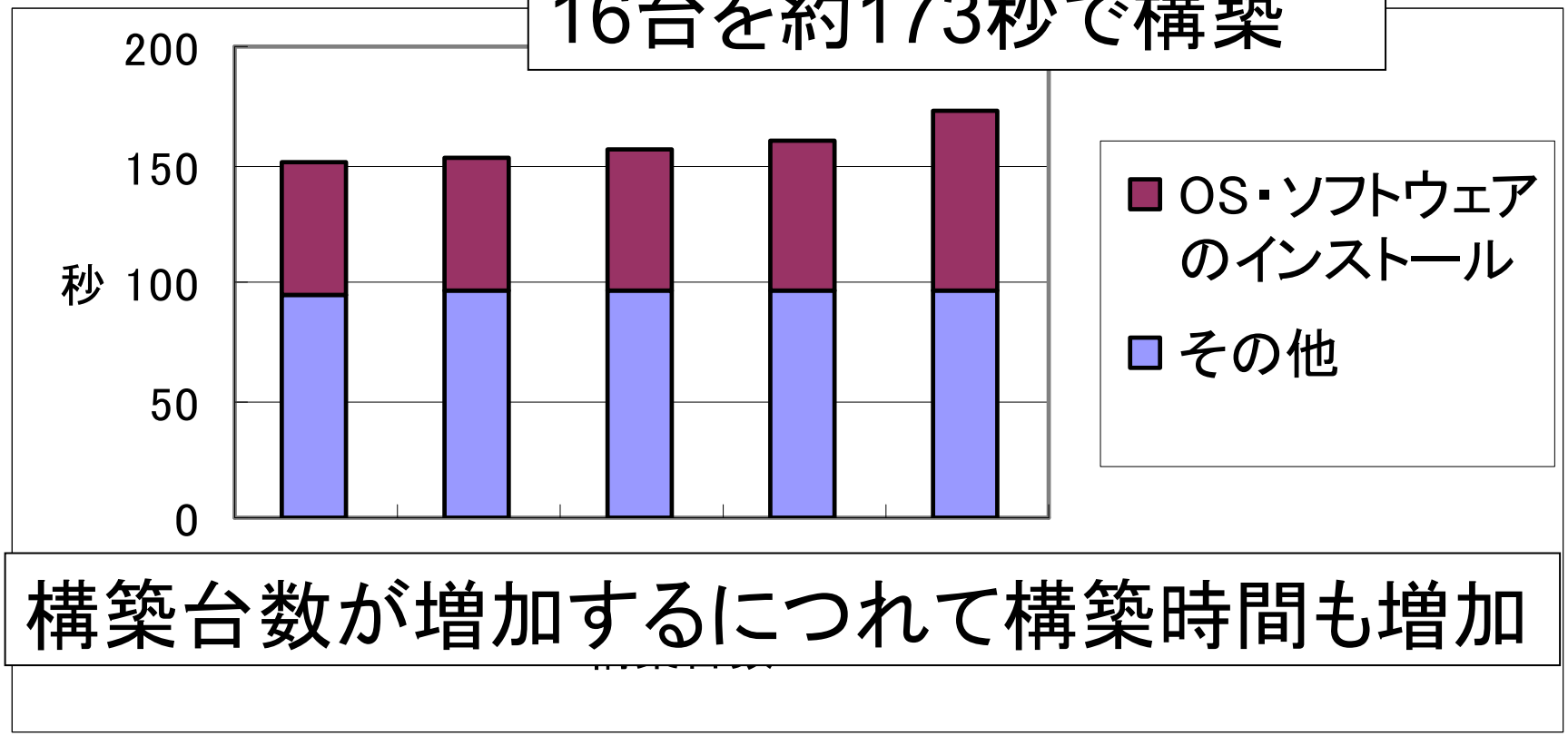
ORE Grid概要


サイト境界線
ユーザ
環境構築機構
①環境構築を依頼
②実行環境のインストールを依頼
③仮想計算機を起動
インストールサーバ
④ブートイメージをコピー、展開
計算機群
パッケージサーバ
⑤パッケージの取得、インストール
仮想計算機群
ジョブ実行機構
⑥ジョブを実行

プロトタイプシステム構築時間

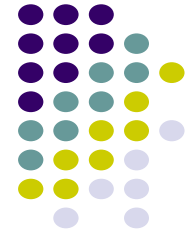
相同性検索ツールBlastのジョブ実行環境を構築(1～16台)

16台を約173秒で構築

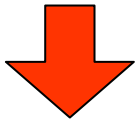


構築台数が増加するにつれて構築時間も増加

プロトタイプシステムの問題点



- 構築に要する時間が長い
  - ジョブを実行するユーザから見ると単なるオーバーヘッドでしかない
  - より高速な構築が求められる場合も存在
- 構築台数における構築時間のスケーラビリティが低い
  - 計算機セットアップのパッケージインストールによるパッケージサーバーへの負荷集中が主な原因

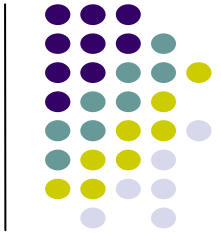


スケーラブルで高速な環境構築機構が必要

本研究の目的と成果

- 目的
  - ジョブ実行環境構築システムのスケーラビリティの向上と高速化
- 成果
  - 一度構築したイメージの再利用(キャッシュ)による高速化
  - キャッシュ使用の有効性を判断するための構築時間変化を定式化したコストモデルの作成とその検証

提案

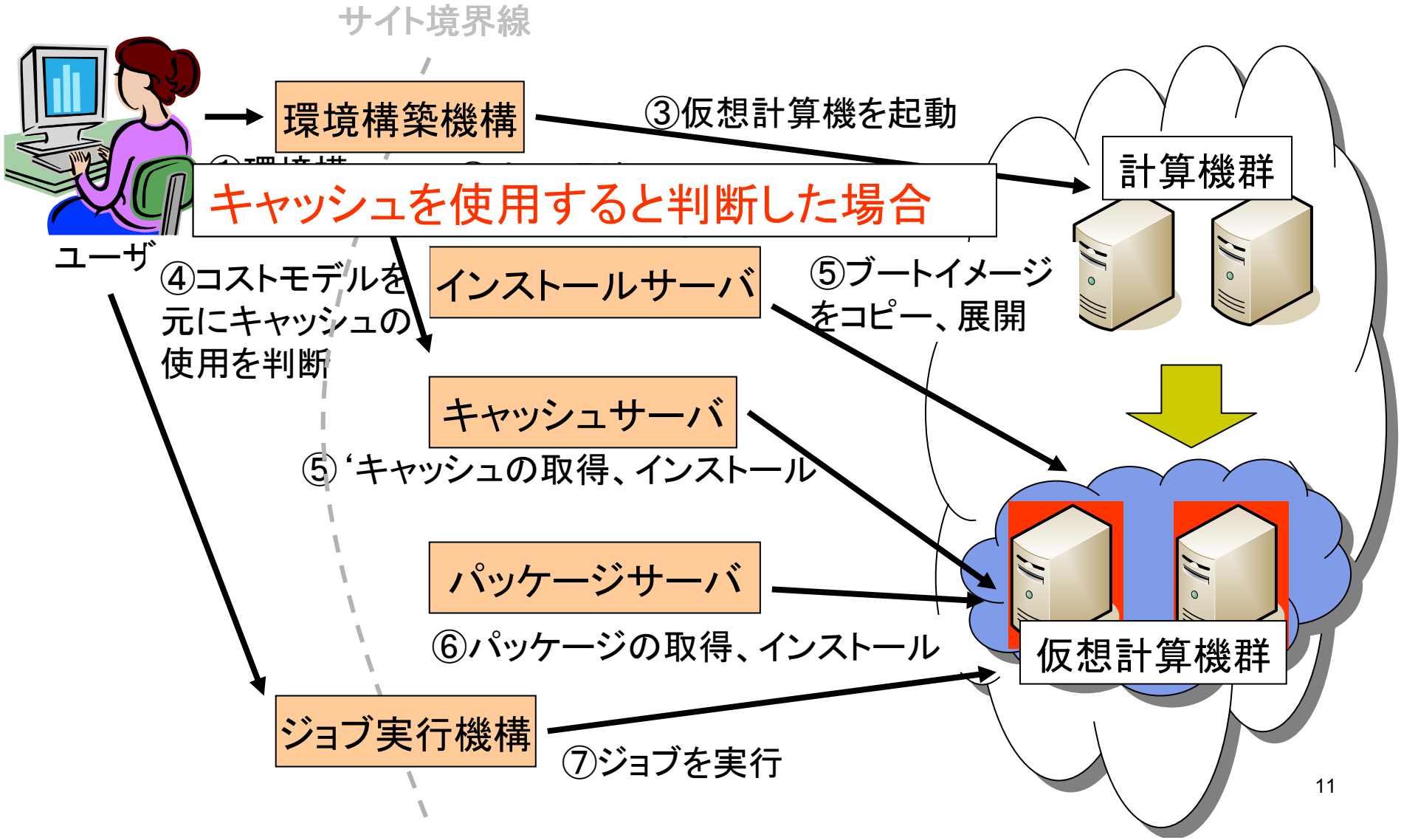


- 一度構築した仮想計算機のイメージを保存(キャッシュ)
  - 同じアプリケーション環境を構築する場合キャッシュを使用
    - パッケージサーバへの負荷の集中が削減
- キャッシュを使用することによる削減時間を定式化(コストモデル)
  - キャッシュの使用、不使用をコストモデルを用いて自動的に決定
    - キャッシュを使用することが常に高速化につながるとは限らない

キャッシュ形式の検討

- 仮想計算機の仮想ディスクイメージ
  - 利点
    - ディスクのパーティション設定などを行う必要なし
  - 欠点
    - 使用しているデータ容量如何にかかわらず仮想ディスクのサイズがキャッシュのサイズになる
    - 仮想計算機ごとにキャッシュを用意する必要
- /binや/varなどのブートイメージを圧縮したもの
  - 利点
    - キャッシュ形式が仮想計算機によらない
  - 欠点
    - パーティション設定などの副次的メリットが存在しない
- 本研究では後者を使用
  - キャッシュ形式が仮想計算機に依存しないため再利用が容易

提案システムの環境構築手順



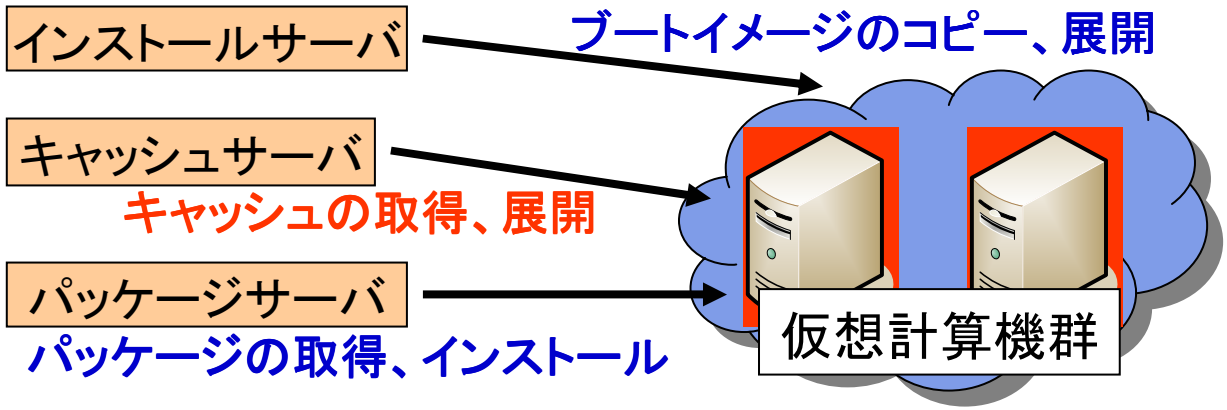
コストモデル

- キャッシュを使用することによる構築時間の変化を定式化
  - キャッシュを使用することにより削減される時間
    - *CP(BootImage)*：ブートイメージのコピー時間
    - *UNPACK(BootImage)*：ブートイメージの展開時間
    - *INSTALL(Package)*：パッケージの取得、インストール時間
  - キャッシュを使用することにより増加する時間
    - *CP(Cache)*：キャッシュの取得時間
    - *UNPACK(Cache)*：キャッシュの展開時間

CP(BootImage)+ UNPACK(BootImage)+ INSTALL(Package)
> *CP(Cache) + UNPACK(Cache)*
である場合キャッシュの使用が有効と判断

キャッシュを使用しない場合の構築手順

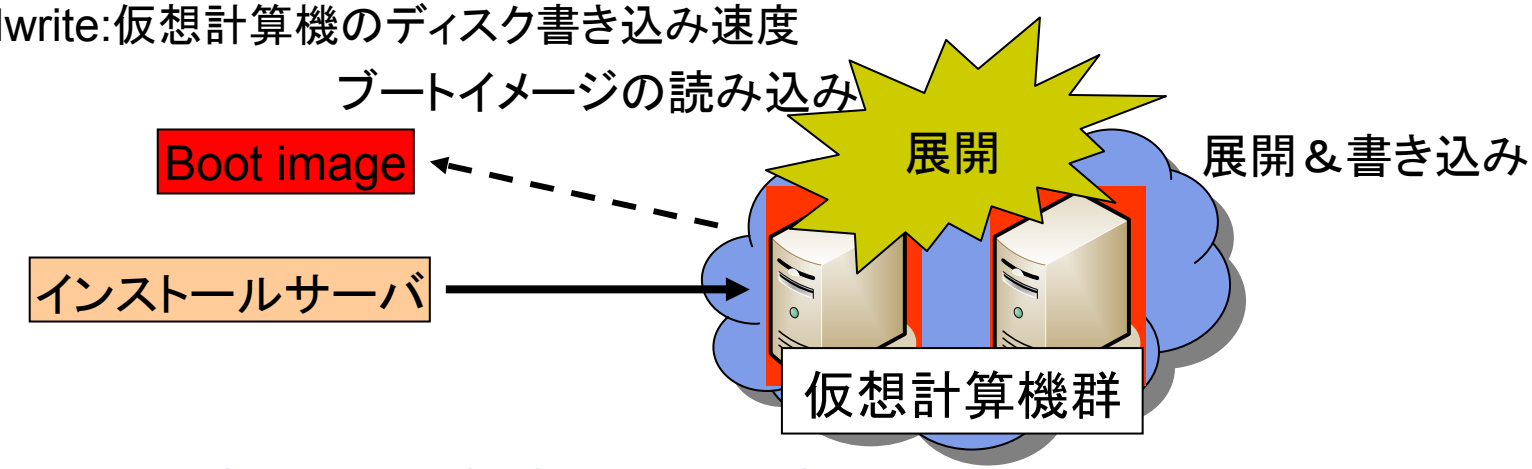
キャッシュを使用する場合の構築手順



プロトタイプにおけるコストモデルの作成：ブートイメージのコピー、展開時間

CP(BootImage) + UNPACK(BootImage)+ INSTALL(Package)

- ブートイメージを読み込む時間(*CP(BootImage)*)
  - Bsize/Min(Nread,Nnetwork)
    - Bsize:ブートイメージのサイズ
    - Nread:インストールサーバのディスク読み込み速度
    - Nnetwork:インストールサーバと仮想計算機間のバンド幅
- ブートイメージを展開し書き込む時間(*UNPACK(BootImage)*)
  - Bunpacksize/VMwrite
    - Bunpacksize：ブートイメージの展開後のサイズ
    - VMwrite:仮想計算機のディスク書き込み速度

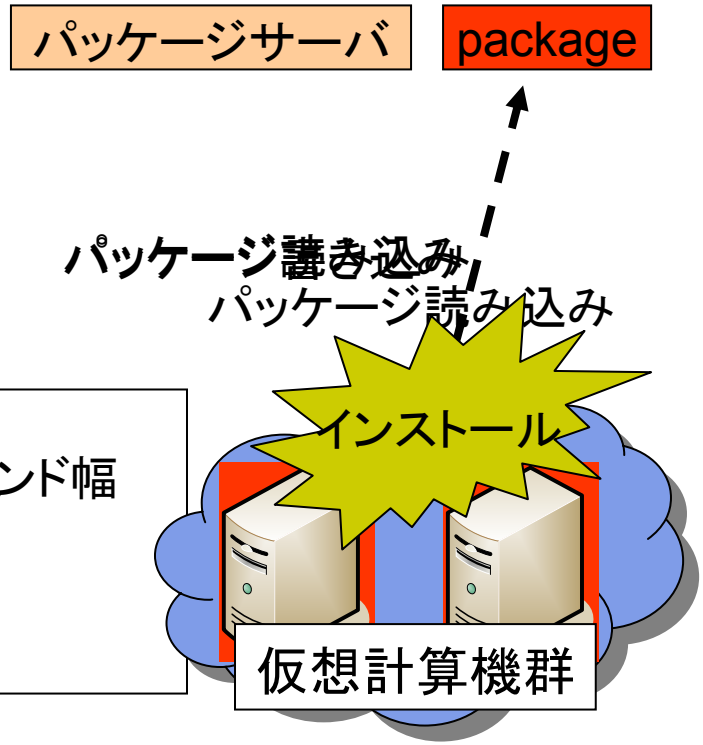


CP(BootImage) + UNPACK(BootImage)
→Bsize/Min(Nread,Nnetwork) + Bunpacksize/VMwrite

プロトタイプにおけるコストモデルの作成: パッケージの取得、インストール時間

CP(BootImage)+ UNPACK(BootImage)+ INSTALL(Package)

- パッケージインストール時間(*INSTALL(Package)*)
  - パッケージのパッケージサーバから読み込む時間
    - Psize/Min(Pread,Pnetwork)
  - パッケージを仮想計算機に書き込む時間
    - Psize/VMwrite
  - パッケージを仮想計算機から読み込む時間
    - Psize/VMread
  - パッケージをインストールする時間
    - Punpacksize/VMwrite



Pread	:パッケージサーバの読み込み速度
Pnetwork	:パッケージサーバと仮想計算機間のバンド幅
VMread(write)	:仮想計算機の読み込み(書き込み)速度
Psize	:パッケージサイズ
Punpacksize	:パッケージの展開後のサイズ

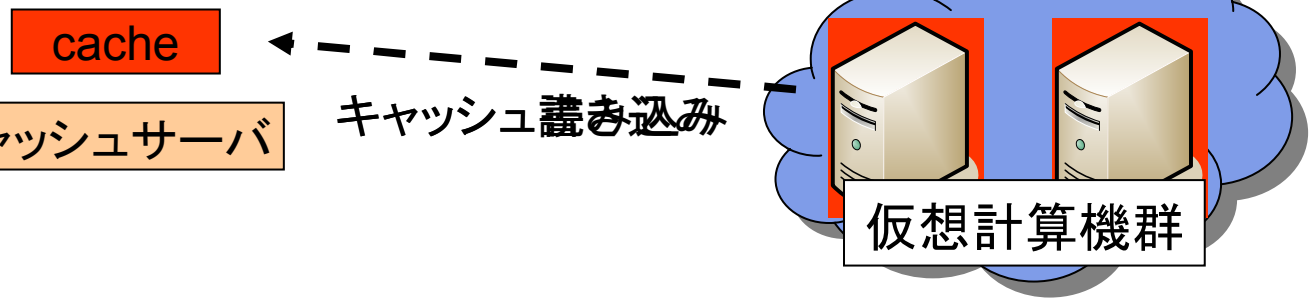
INSTALL(Package)
→Psize/Min(Pread, Pnetwork) + Psize/VMwrite + Psize/VMread + Punpacksize/VMwrite

プロトタイプにおけるコストモデルの作成: キャッシュの取得時間

CP(Cache) + *UNPACK(Cache))*

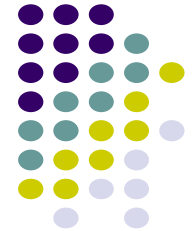
- キャッシュ取得時間(*CP(Cache)*)
  - キャッシュサーバからキャッシュを読み込む時間
    - Csize/Min(Cread,Cnetwork)
  - 読み込んだキャッシュを仮想計算機のディスクに書き込む時間
    - Csize/VMwrite

Cread　　:キャッシュサーバの読み込み速度
Cnetwork:キャッシュサーバと仮想計算機間のバンド幅
VMwrite　:仮想計算機の書き込み速度
Csize　　:キャッシュのサイズ



CP(Cache)
→Csize/Min(Cread, Cnetwork) + Csize/VMwrite

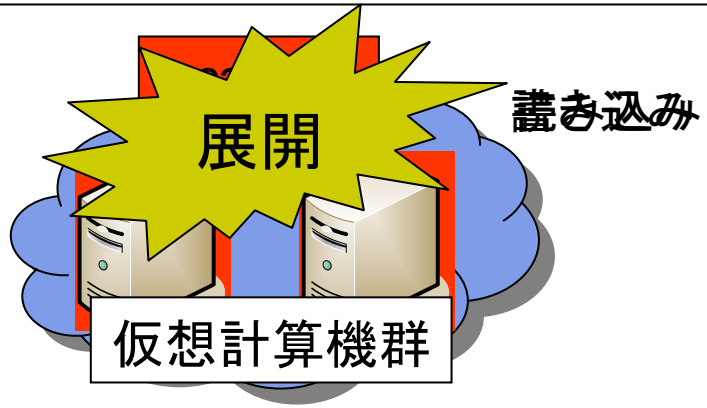
プロトタイプにおけるコストモデルの作成: キャッシュの展開時間



CP(Cache) + *UNPACK(Cache)*

- キャッシュ展開時間(*UNPACK(Cache)*)
  - キャッシュを仮想計算機のディスクから読み込む時間
    - Csize/VMread
  - キャッシュを展開し仮想計算機のディスクへ書き込む時間
    - Cunpacksize/VMwrite

VMread(write)	:仮想計算機の読み込み(書き込み)速度
Csize	:キャッシュのサイズ
Cunpacksize	:キャッシュの展開後のサイズ

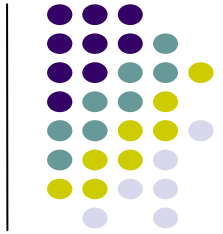


UNPACK(Cache)
→Csize/VMread + Cunpacksize/VMwrite

プロトタイプにおけるコストモデルの作成

- 以下の値を解析することにより構築時間の予測が可能
  - インストールサーバと仮想計算機間バンド幅
  - パッケージサーバと仮想計算機間バンド幅
  - キャッシュサーバと仮想計算機間バンド幅
  - 仮想計算機ディスクバンド幅
  - パッケージサーバのディスクバンド幅
  - インストールサーバのディスクバンド幅
  - キャッシュサーバのディスクバンド幅
  - インストールするパッケージサイズと展開後のサイズ
  - 対応するキャッシュサイズと展開後のサイズ
- 予測値と実測値を比較し検証

評価



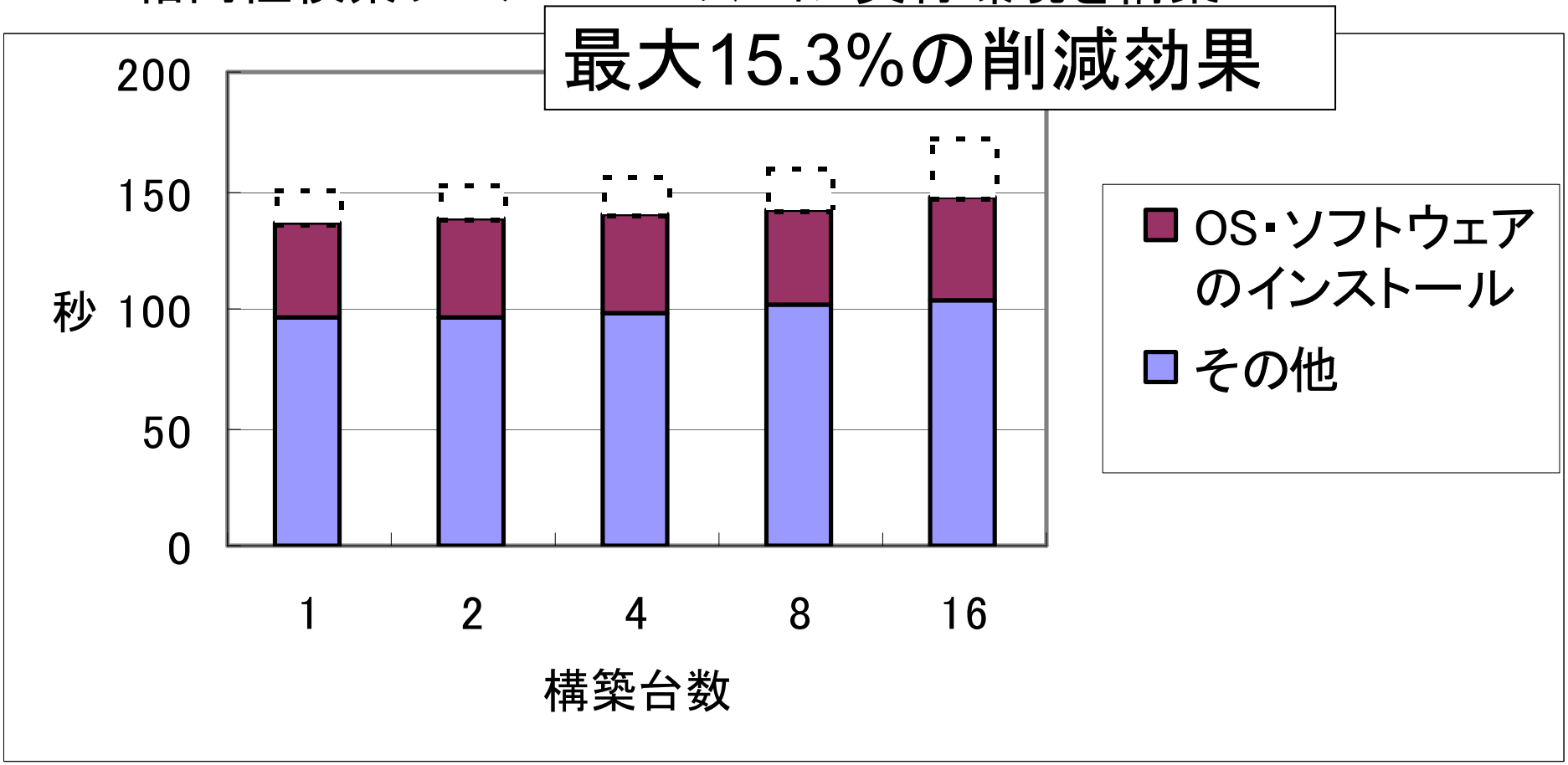
- 評価項目
  - キャッシュを使用することによる構築時間の変化
  - 構築したコストモデルと実測値との比較、検討
- 実装したシステムを用いてジョブ実行環境を構築
  - HDD20GB、memory256MB、OS:Debian、カーネル2.4.31
- 評価環境は松岡研究室PrestoIIIクラスタ

評価環境

CPU	AMD Opteron™ Processor250(2.4GHz)×2
メモリ	PC2700 2GB
OS	Debian GNU/Linux sarge
ネットワーク	Gigabit Ethernet
仮想計算機	VMware5.0(linux版)Workstation

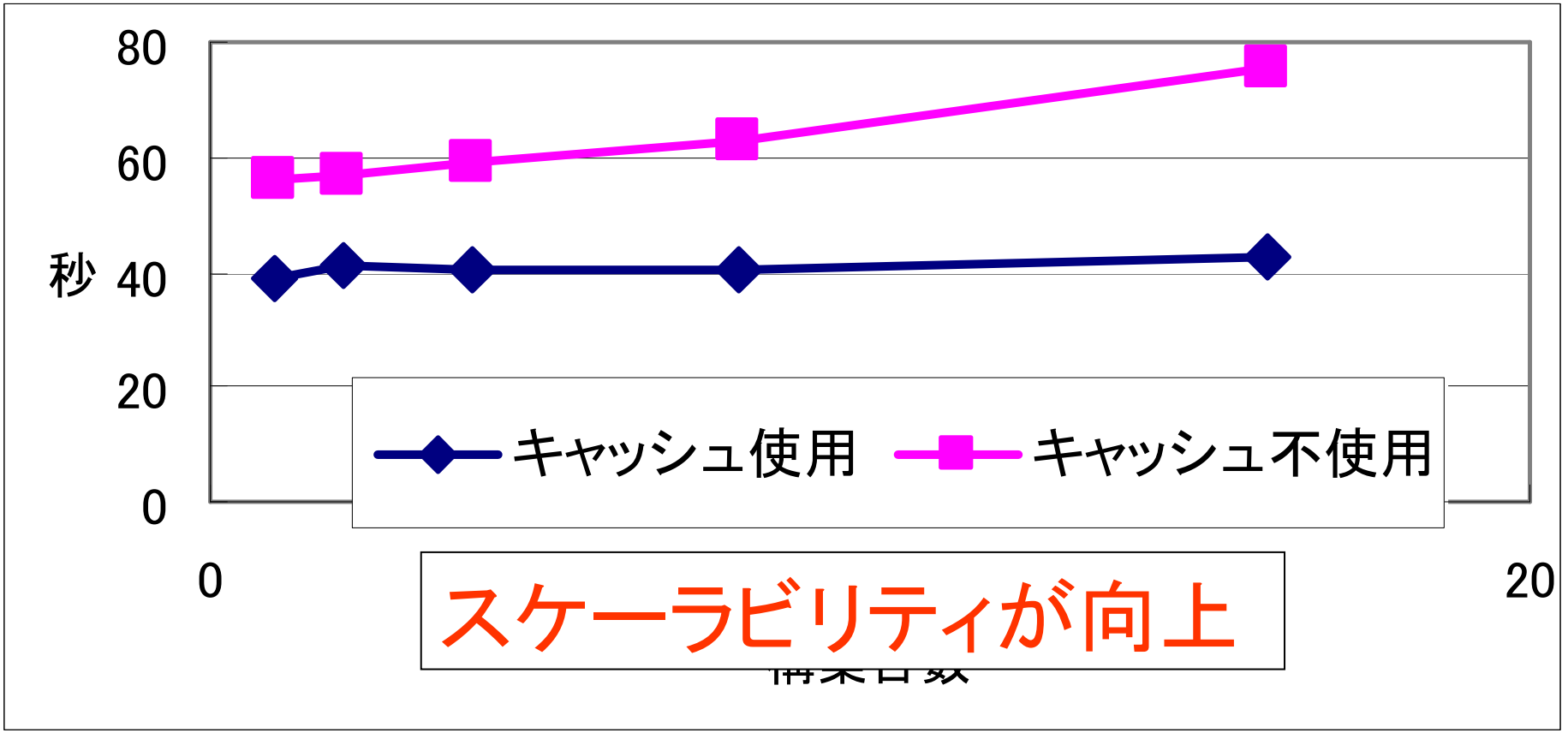
評価：構築時間の変化

相同性検索ツールBlastのジョブ実行環境を構築



評価:スケーラビリティの検証

Blastジョブ実行環境の仮想計算機セットアップ時間の推移
(1～16台)

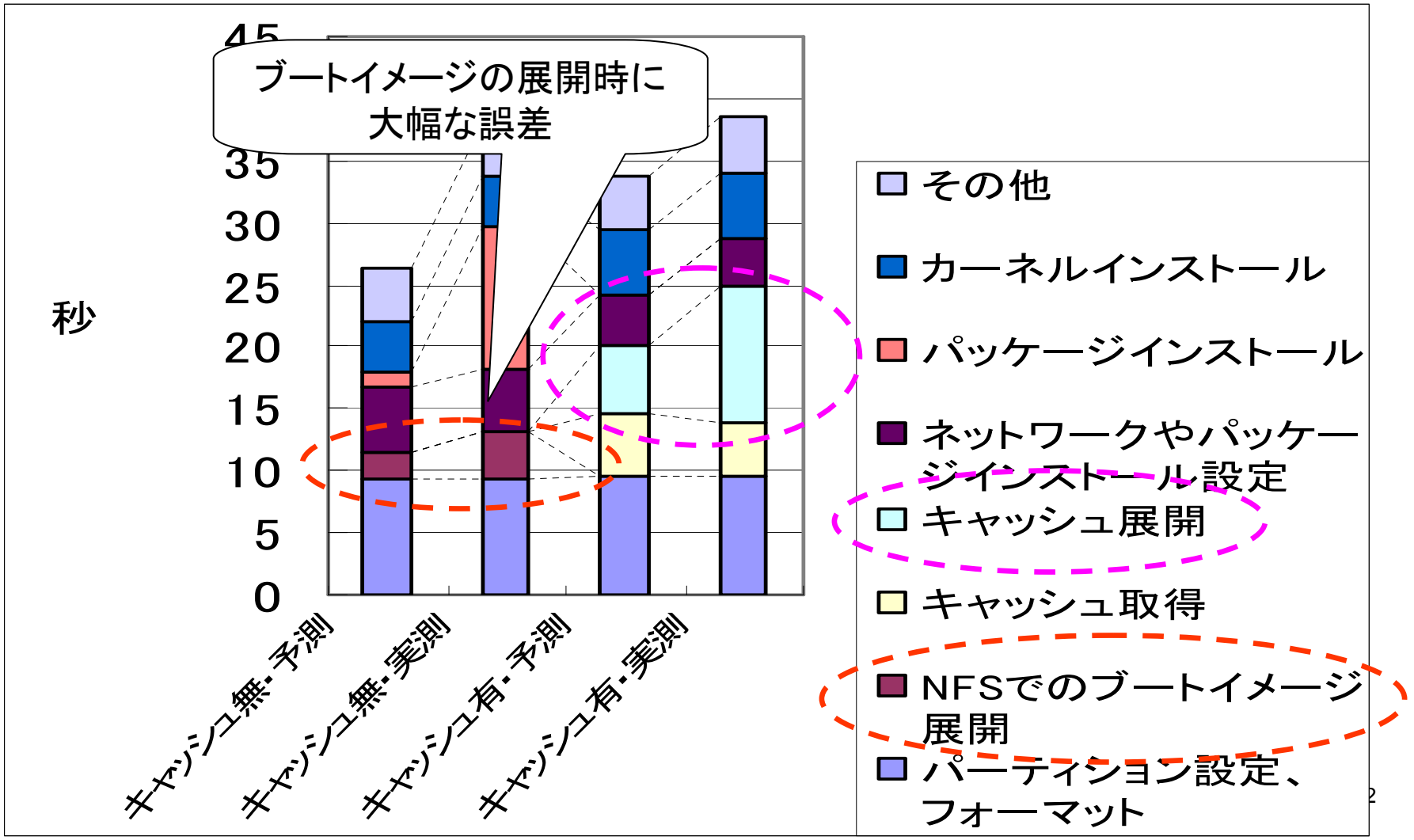


評価: コストモデルの検証

- さまざまな条件の下で環境構築を行い、コストモデルを検証
- 代表として以下の環境でBLASTジョブ実行環境を構築
  - 計測にはiperfとbonnie+を使用

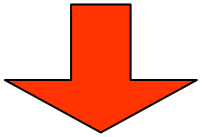
インストールサーバと仮想計算機間バンド幅	112MB/sec
パッケージサーバと仮想計算機間バンド幅	112MB/sec
キャッシュサーバと仮想計算機間バンド幅	112MB/sec
仮想計算機ディスクバンド幅	read:64.6MB/sec write:69.8MB/sec
パッケージサーバのディスクバンド幅	read:64.0MB/sec
インストールサーバのディスクバンド幅	read:65.4MB/sec
キャッシュサーバのディスクバンド幅	read:46.3MB/sec
インストールするパッケージサイズ	取得時:11.854MB 展開時44.38MB
対応するキャッシュサイズ	取得時:135.1MB 展開時250.5MB

評価: コストモデルの検証



考察: ブートイメージ展開時の予測値と実測値の誤差

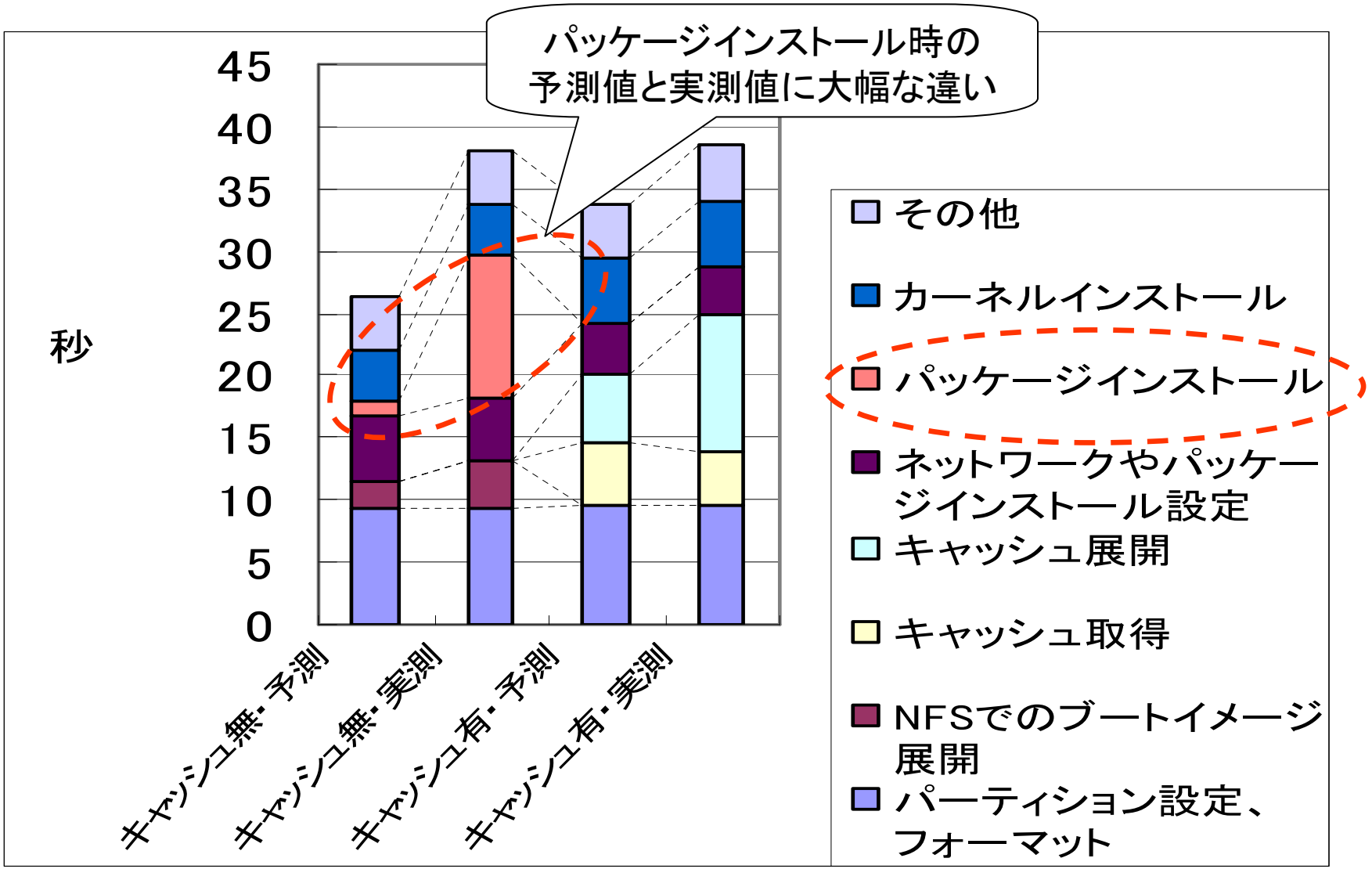
- ブートイメージとは/binや/varなどの起動に必要なイメージを圧縮したもの
  - 一般的な圧縮ファイルと比較してファイルやディレクトリが非常に多い
  - ファイルやディレクトリを作成するコストにより誤差が生じていると推測



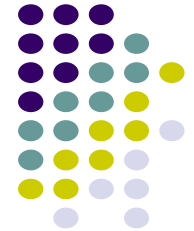
今後の課題

- ファイル数の展開時間への影響を調査し、コストモデルへ組み込むかどうかを検証

評価: コストモデルの検証



考察: パッケージインストール時の予測値と実測値の誤差



- パッケージ取得、インストール以外のオーバーヘッドが存在
  - パッケージインストール時にローカルデータベースへアクセス
    - 指定したアプリケーションがインストールされていないかどうかを検索
  - インストーラー起動のみに約2.5秒必要

今後の課題

- パッケージインストールの初期コストを加えたコストモデルの作成
- インストーラーの特性に合わせたコストモデルの作成

関連研究: VMplants [I. Krsul et al, '04]

- ユーザのジョブ実行環境を仮想計算機を用いて提供するシステム
- ゴールデンイメージを用いた環境の構築が可能
  - よく使われるインストールや環境構成が完了したシステムのスナップショット
  - 仮想計算機の起動を高速化
- ゴールデンイメージを用いることによる構築時間の変化については検討がなされていない

関連研究: Virtual Workspaces[K.Keahey et al, '06]

- 仮想計算機上に実行環境を構築するシステム
- 仮想計算機システムとしてXenが使用可能
  - 各計算機へのディスクイメージの転送にGridFTP[W. Allcock et al, '01]を使用
    - ディスクイメージの転送を高速化
- 構築する環境を指定することが不可能
  - ユーザが各ノードでアプリケーションをインストールする必要

まとめ・今後の課題

- まとめ
  - ジョブ実行環境構築システムの高速化のために一度構築したイメージの再利用(キャッシュ)する手法を提案
  - キャッシュ使用の有効性を判断するための構築時間変化を定式化したコストモデルの作成と検証した
- 今後の課題
  - コストモデルの再構成とシステムへの組み込み
  - キャッシュシステムの拡張
  - ジョブ実行環境構築システムの拡張

謝辞

- 本研究の一部は独立行政法人新エネルギー・産業技術開発機構 基盤技術研究促進事業(民間基盤技術研究支援制度)の一環として委託を受け実施している「大規模・高信頼サーバの研究」の成果である