仮想クラスタのステートレス化のための Rocks 5ディスクレス化機構

小川宏高　中田秀基　広渕崇宏
伊藤智　関口智嗣
（産総研）

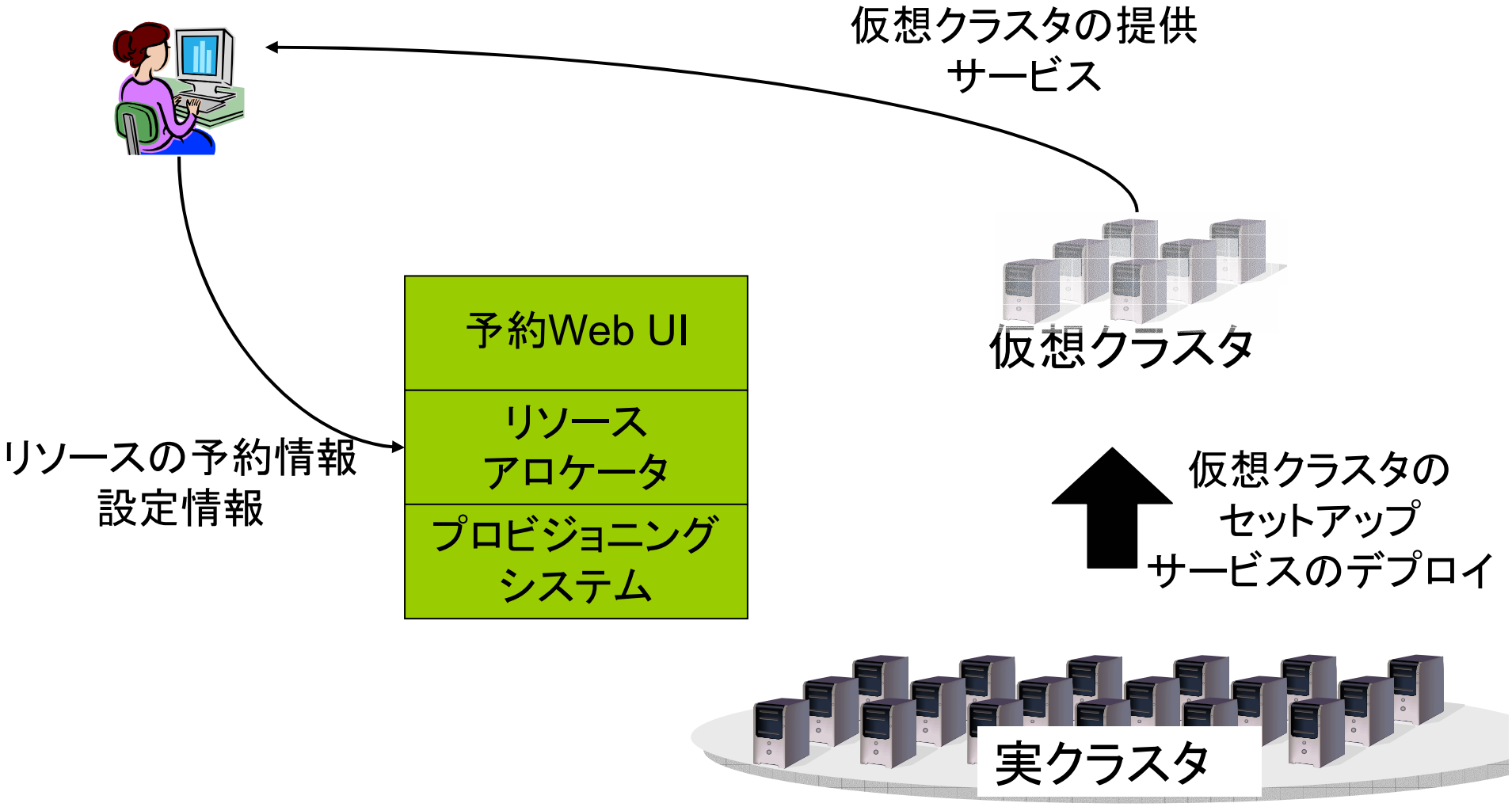
背景

- 計算機センター・データセンター
  - 稼働率の向上
  - 設備コスト・運用コストの削減
- →仮想化技術
  - 運用の自由度〇
  - 資源の稼働率〇
  - 設備コスト〇
  - 運用の複雑さ・手間×
- →仮想クラスタ
  - 仮想計算機システム全体(計算機、ネットワーク、ストレージなど)を統合的に管理
  - OS、ミドルウェア、アプリケーションなどのプロビジョニング
  - 運用の複雑さや手間を削減〇

Grivon仮想クラスタ管理システム

- 仮想クラスタを「ユーティリティ的」に構築・実現
  - オンデマンドかつフレキシブルに、ユーザの必要とする仮想クラスタを構成・提供
- 構成要素
  - 予約Webインタフェースおよびサービス
    - 利用する計算機資源の量(CPU, メモリー, ディスク容量など)、構成情報、占有時間などを指定
  - 仮想計算資源アロケータ
    - 予約情報に基づき、VMware Server 1.0による仮想計算機、VLAN、iSCSIストレージなど仮想計算資源を生成・確保
  - 仮想計算資源プロビジョニングシステム
    - アロケートされた仮想計算資源に対して、NPACI Rocks Toolkitを用いてOS, アプリケーションなどを自動的に配備
    - Rollと呼ばれるメタパッケージが充実している
      → 主要な科学技術計算ソフトウェアへの対応

Grivon概要


仮想クラスタの提供
サービス
予約Web UI
リソース
アロケータ
プロビジョニング
システム
リソースの予約情報
設定情報
仮想クラスタ
仮想クラスタの
セットアップ
サービスのデプロイ
実クラスタ

関連研究

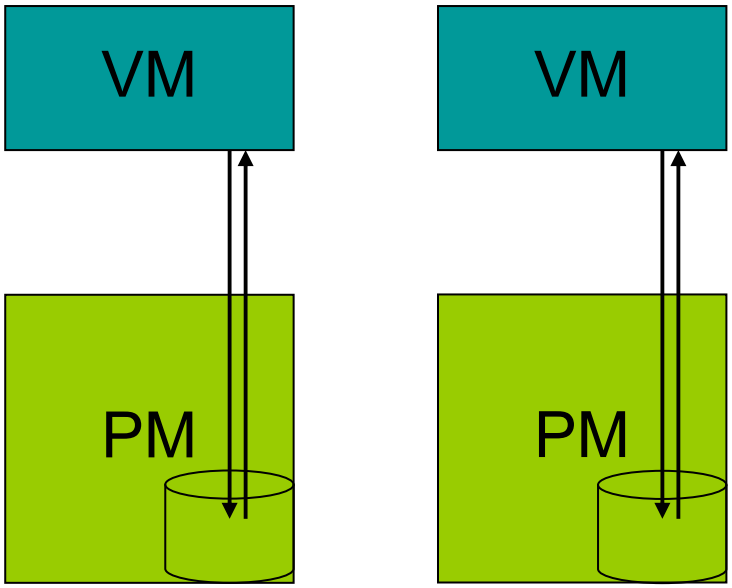
- ORE Grid [高宮 ’06, Nishimura ‘07]
  - クラスタインストールツールlucieを利用
  - 超高速ノードインストール
- Virtual workspace [Keahey ’06]
  - Globusプロジェクトの一環
  - Web Servicesベースのインターフェイスで実行環境を構成し、そこにジョブを投入
  - 基本的に、ジョブ単位、ノード単位
- OSCARによるXenクラスタ [Vallee ‘06]
  - OSCAR (Rocksに相当するクラスタデプロイツール)
- Cisco VFrame
  - InfinibandネットワークとSAN、専用スイッチを用いてストレージとネットワークを仮想化
  - 計算機は仮想化されていない
  - 非常に高価な専用ハードウェアが必須

本研究の目的

- 仮想クラスタのストレージ問題
  - 容量
  - 性能
  - 管理の自由度・容易性(c.f., 遠隔ライブマイグレーション)
- 仮想クラスタシステムのステートレス化
  - ストレージの問題を物理クラスタから分離
    - 仮想マシンをディスクレス化
    - 仮想マシンをサービスする物理マシンもディスクレス化
  - 容量、性能、管理上の問題の解決を図る
- 実装
  - Rocks version 5への汎用的な実装
    - VMMの実装、Grivonの実装に依存しない
  - Rocksの管理下の任意のノードをディスクレス化できる

仮想クラスタのストレージ

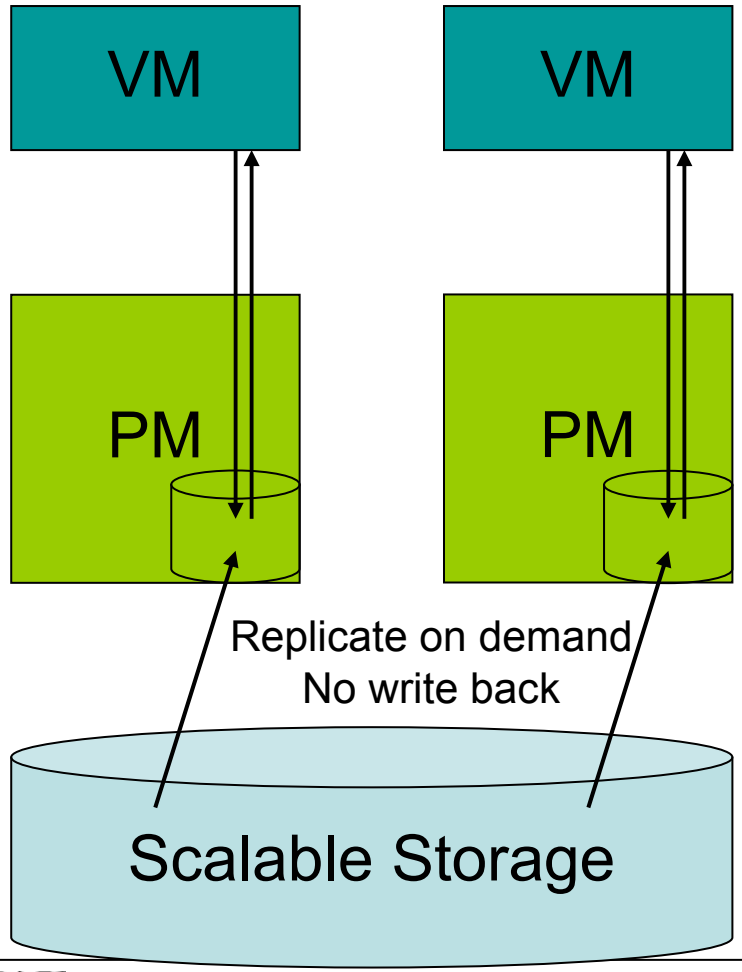
- 物理マシンのディスク装置を利用
- 管理対象が分散
- 負荷分散や障害対策のためにファイルシステムをマイグレートするのが困難
- 性質
  - I/O性能 ○ / △
  - 容量　　×
  - 管理性　×



単一(または少数)のディスク装置に制約される

仮想クラスタのストレージ

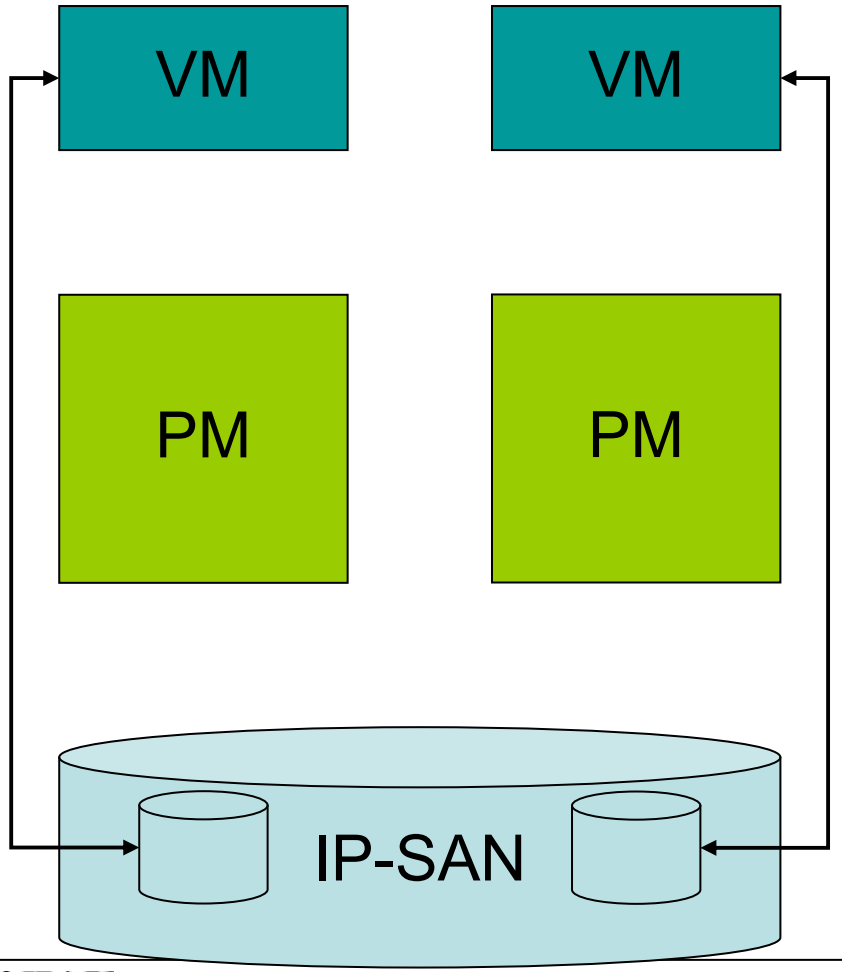
- Amazon EC2
  - ストレージクラスタをバックエンドに用意
  - オンデマンドでVM用ストレージをPM上に複製
  - Write backしない = ファイルシステムとしての透明性は捨てる
- 性質
  - I/O性能 ○ / △
  - 容量　　○
  - 管理性　○ / △



エンジニアリング的に優れた解のひとつ

仮想クラスタのストレージ

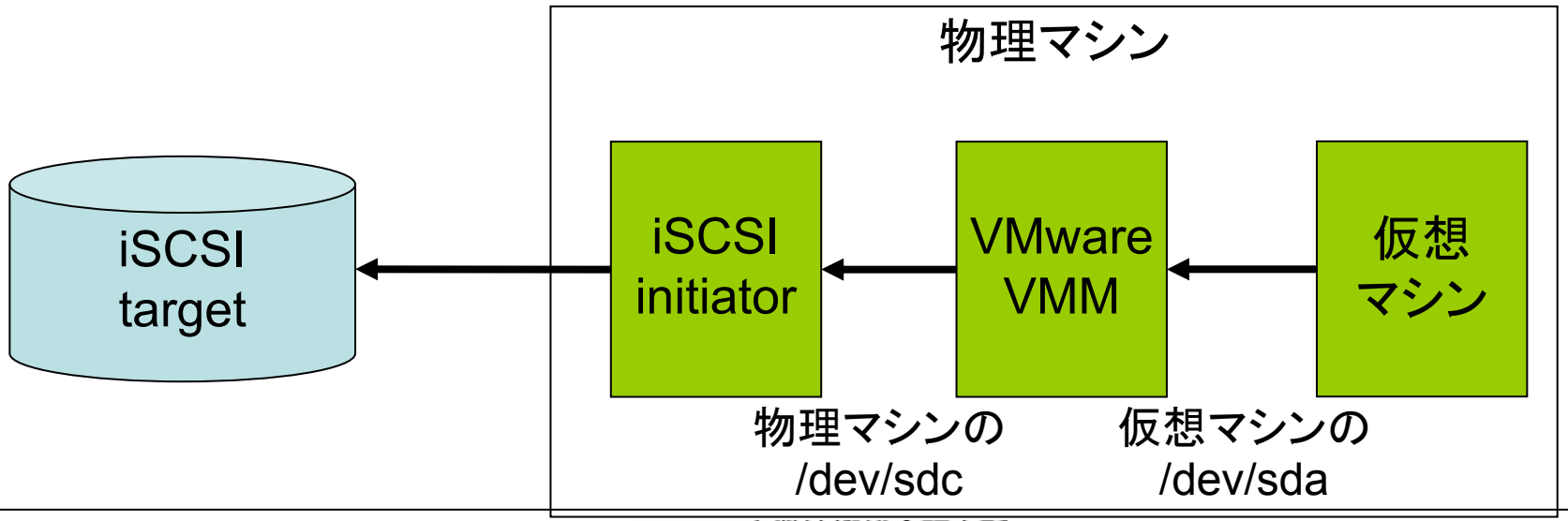
- IP-SAN (iSCSI)を利用する
  - ストレージクラスタをバックエンドに用意
  - VMもしくはPMがiSCSIイニシエータとなってIP-SAN上のストレージをファイルシステムとして利用
- 性質
  - I/O性能 ○ / △
  - 容量　　○
  - 管理性　○



我々が開発しているGrivonもこの方法を採用

Grivonでの実装

- iSCSIを仮想クラスタのストレージとして利用
  - 各仮想マシンのストレージは、物理マシンから分離される
    →餅は餅屋
  - 大容量・並列読み書き性能・耐故障性
    →複数のRAIDコントローラを持つストレージノードのクラスタ化によって実現
  - (ライブ)マイグレーション
    →iSCSI target間での(ライブ)リプリケーションによって実現(できる)

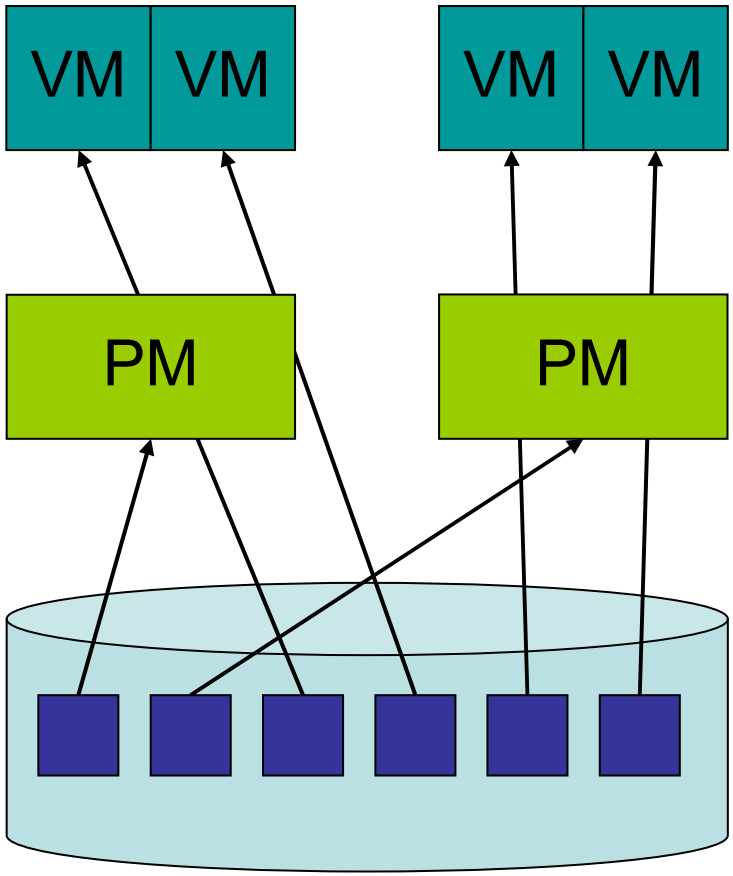


Grivonでの実装(続)

- 物理マシンのディスク装置を、仮想マシンのホストのストレージとして利用する
  - 物理ディスクが分散するため、管理性が低い
  - ファイルシステムをマイグレートするのが困難
    - 障害対策
    - 他の業務にホスト計算機を開放
  - 物理ディスクを仮想クラスタに占有させたくない利用シナリオも存在する
    - 例: 物理マシンの遊休時のみ仮想クラスタに参加させる
- 仮想マシンに提供するストレージは、物理マシンにも利用可能(常考)
- (AIST Super Clusterに限らず)ディスクが壊れて「ラックの肥やし」になっているノードは案外多い

ステートレス仮想クラスタ

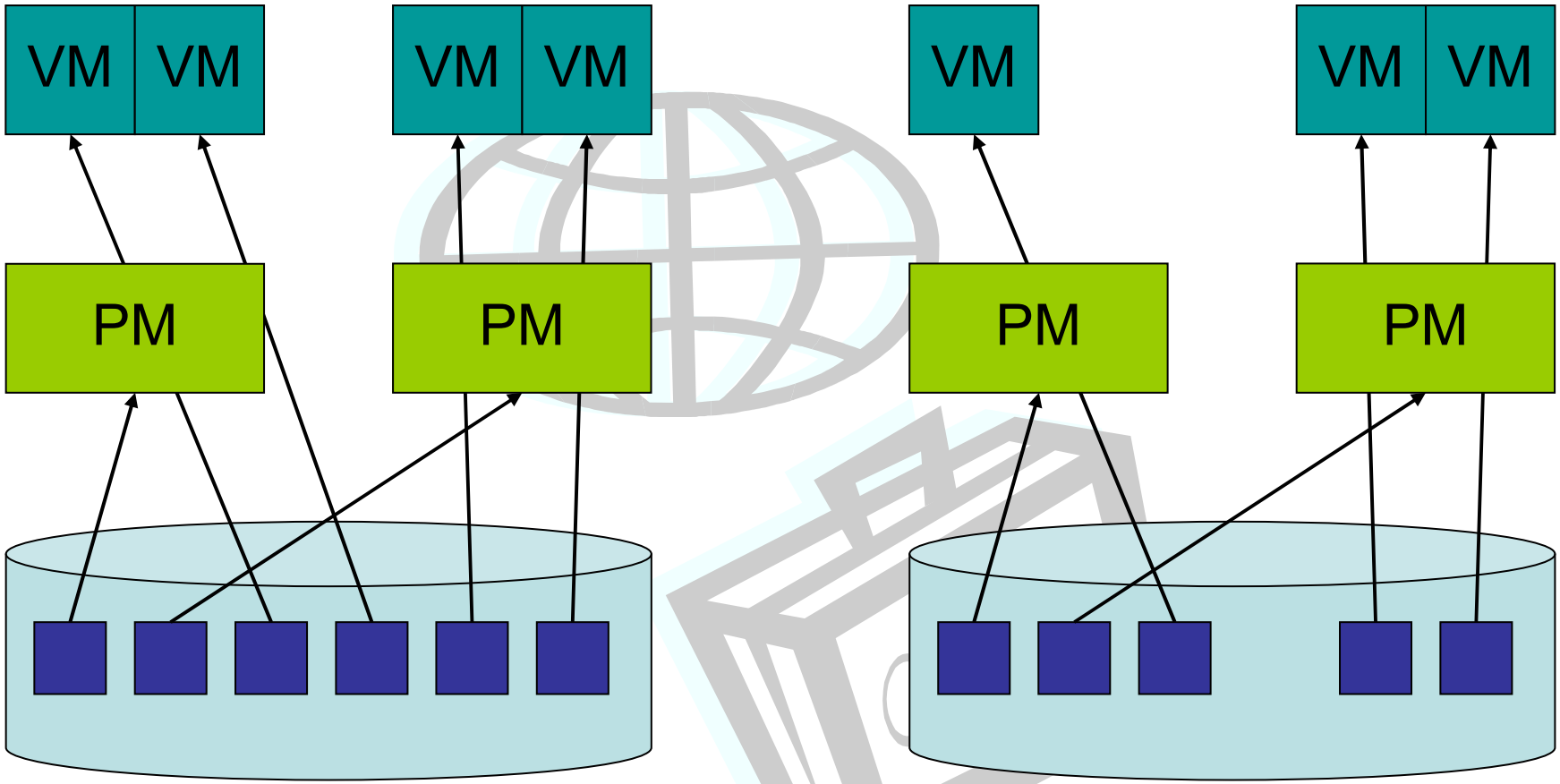
物理ディスクなど捨ててしまえ



- 物理マシンも仮想マシンもディスクレスブート
  - 仮想クラスタシステム全体の、物理ディスクへの依存を一掃
  - ステートをすべてIP-SAN上に格納

ステートレス仮想クラスタ(2)

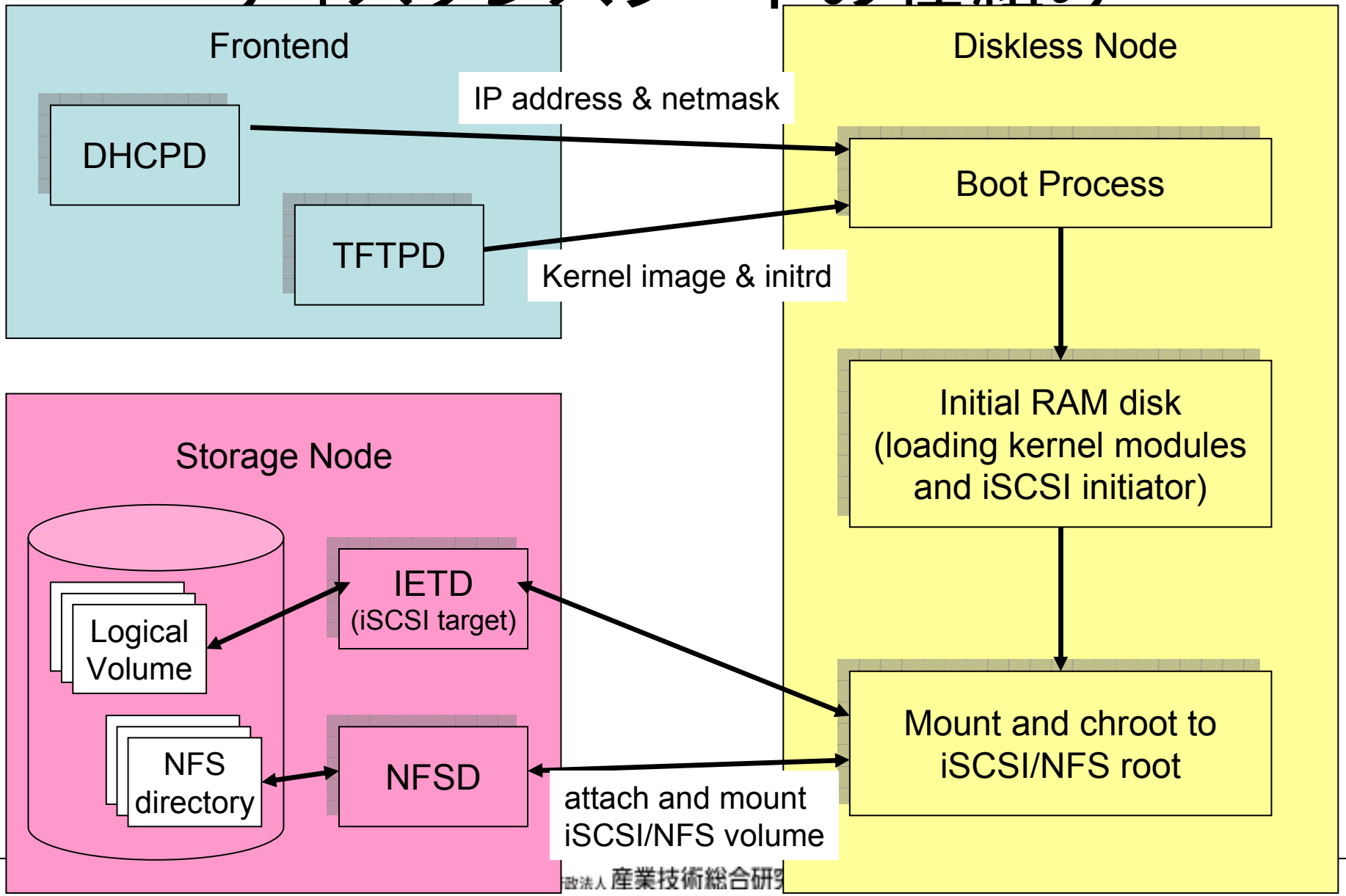
物理マシン・仮想マシンのmobilityを最大化



ディスクレスブート

- PXEブートしたカーネルからリモートストレージをroot mountすることで、ディスクレス状態でマシンを起動
- Linuxの2段階のブートプロセスを利用
  - PXE boot
    - DHCPサーバ、TFTPサーバから得たネットワーク情報、カーネルイメージ、RAMディスクイメージを使ってブート
  - 初期ブートプロセス
    - RAMディスクイメージを一時rootファイルシステムとしてマウント
    - 一時rootファイルシステムを使って、initスクリプトを実行
    - initスクリプトで、iSCSI initiatorを構成してiSCSIデバイスをアタッチ
    - ブロックデバイスを/sysrootにmountして、chroot
  - 本番ブートプロセス
    - 略

ディスクレスブートの仕組み

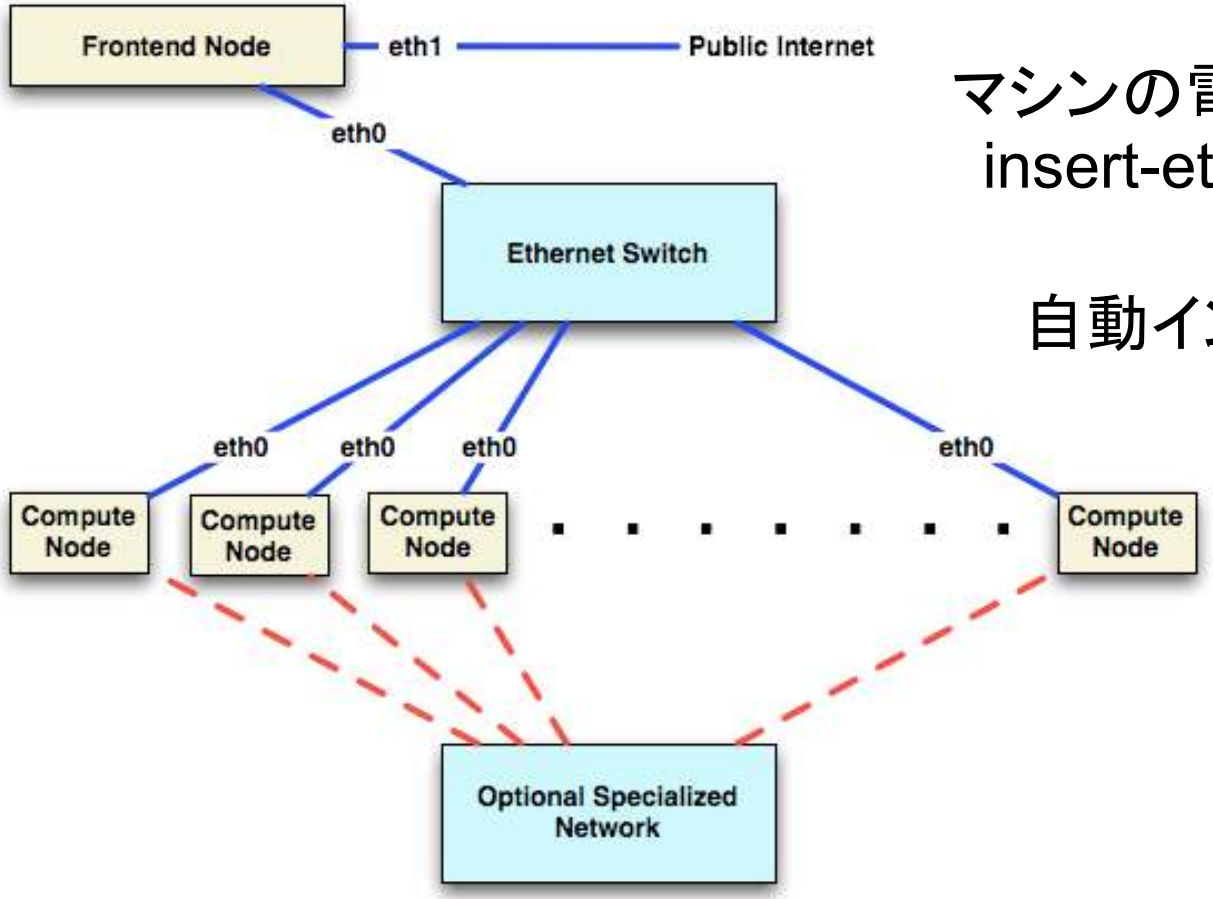


ディスクレス化機構の実装

- Rocks version 5上に実装
  - GrivonはRocksを用いて物理マシン・仮想マシンのセットアップを行う
  - Rocks 5を用いて管理されるクラスタで汎用に利用可能
- 実装内容
  - ディスクレスシステム用のストレージの自動的な確保
  - インストール時・ブート時のディスクレスブートの自動的な制御
    - インストール用のカーネルオプション、initrdなどの生成
    - インストール完了後に次回ディスクレスブート用のカーネルオプション、initrdの再生成

NPACI Rocks

- クラスタインストレーションシステム



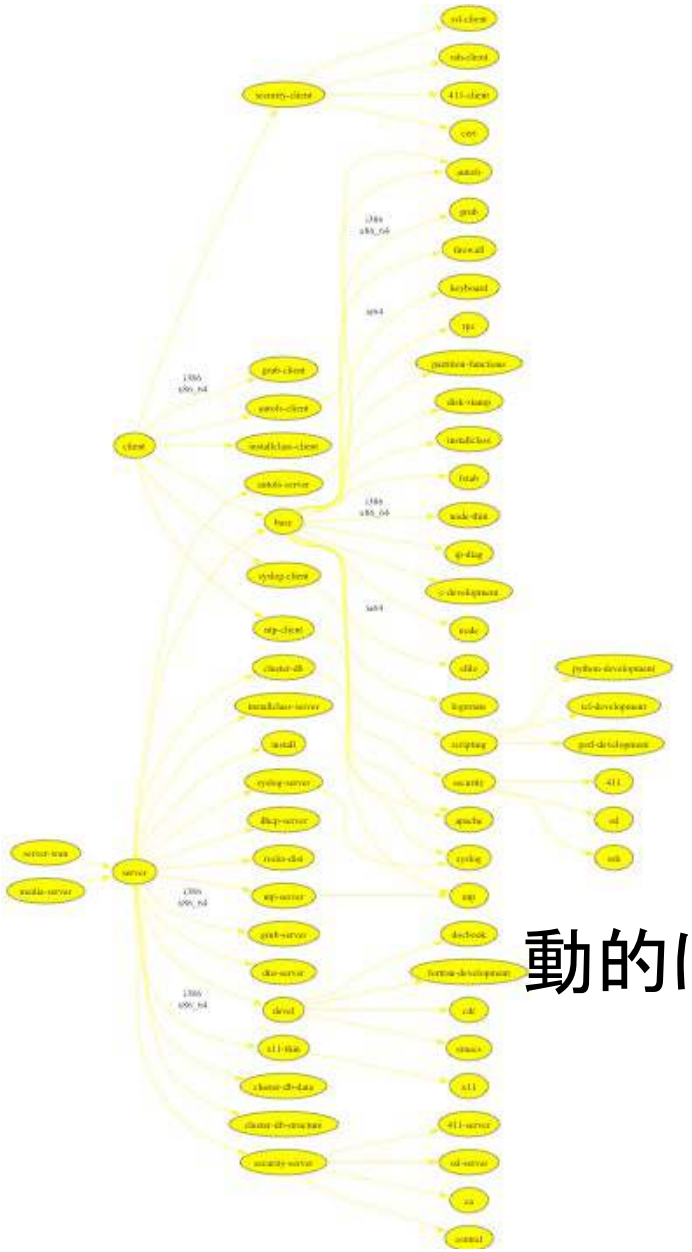
マシンの電源を入れて
insert-ethersを実行
↓
自動インストール

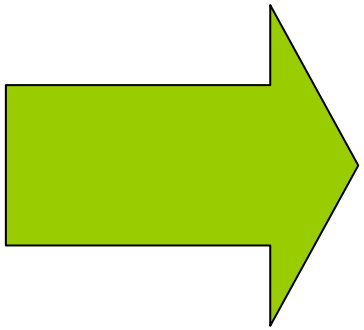
NPACI Rocks (2)

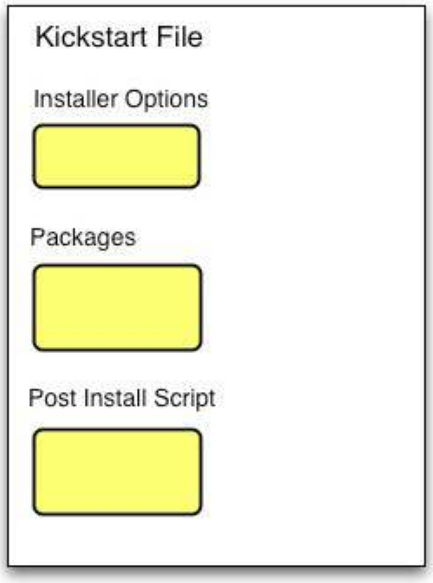
- カスタマイザビリティが特徴
  ＝インストール時のノード、フロントエンドの振る舞いの変更
  - ロール
  - insert-ethersプラグイン
  - ＋α(自前)

NPACI Rocks
クライアントの振る舞いの変更

- ロール
  - Rocksシステムのコア
  - RPMとNode/Graph XMLをまとめてISOにしたもの
    - Node XML: パッケージの詳細とインストール手続き
    - Graph XML: 依存関係を記述
  - ノードのインストール時にロール集合の依存関係にしたがって、kickstartファイルを動的に生成する







ロール集合から
動的にkickstartファイルを生成

NPACI Rocks
クライアントの振る舞いの変更

- ロール
  - Rocksシステムのコア
  - RPMとNode/Graph XMLをまとめてISOにしたもの
    - Node XML: パッケージの詳細とインストール手続き
    - Graph XML: 依存関係を記述
  - ノードのインストール時にロール集合の依存関係にしたがって、kickstartファイルを動的に生成する
  - Anacondaインストーラは動的生成されたkickstartにしたがってインストールする

→したがってインストールされるノードの振る舞いを変更できる

NPACI Rocks
サーバの振る舞いの変更

- insert-ethersプラグイン
  - insert-ethersコマンドを実行してノードの追加を検出したときに実行される
  - ただのpythonスクリプト

  →ノードの追加時のみだが、インストールサーバ側の振る舞いを変更できる
- 自前
  - サーバ側にXMLRPCサービスを用意する
  - Kickstartファイルの中でサーバ側XMLRPCをコールバックするようにロールを記述しておく

  →インストール完了時など任意の時点でのインストールサーバ側の振る舞いを変更できる

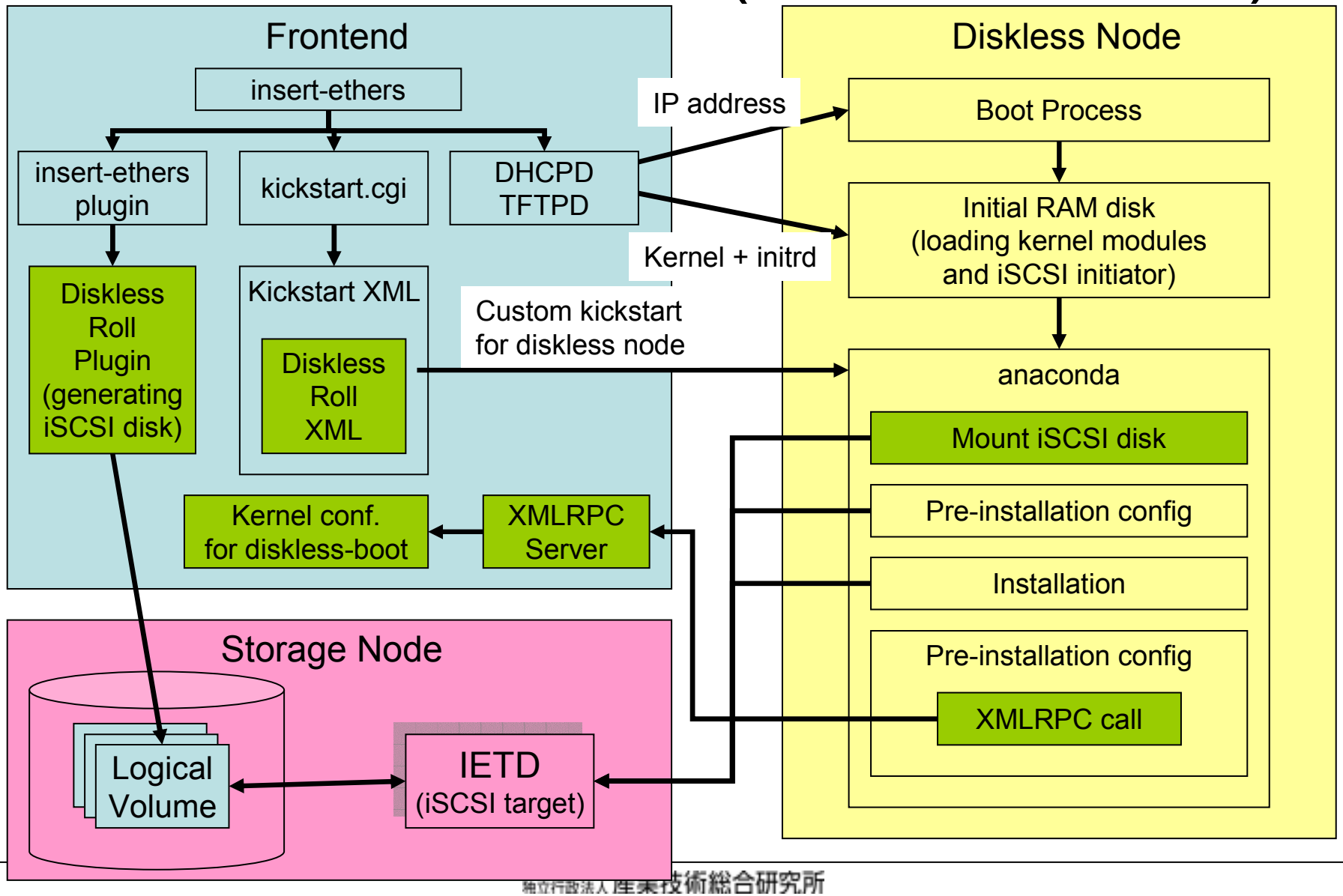
ディスクレス化機構の実装

- Rocks version 5上に実装
  - GrivonはRocksを用いて物理マシン・仮想マシンのセットアップを行う
  - Rocks 5を用いて管理されるクラスタで汎用に利用可能
- 実装内容
  - ディスクレスシステム用のストレージの自動的な確保
  - インストール時・ブート時のディスクレスブートの自動的な制御
    - インストール用のカーネルオプション、initrdなどの生成
    - インストール完了後に次回ディスクレスブート用のカーネルオプション、initrdの再生成

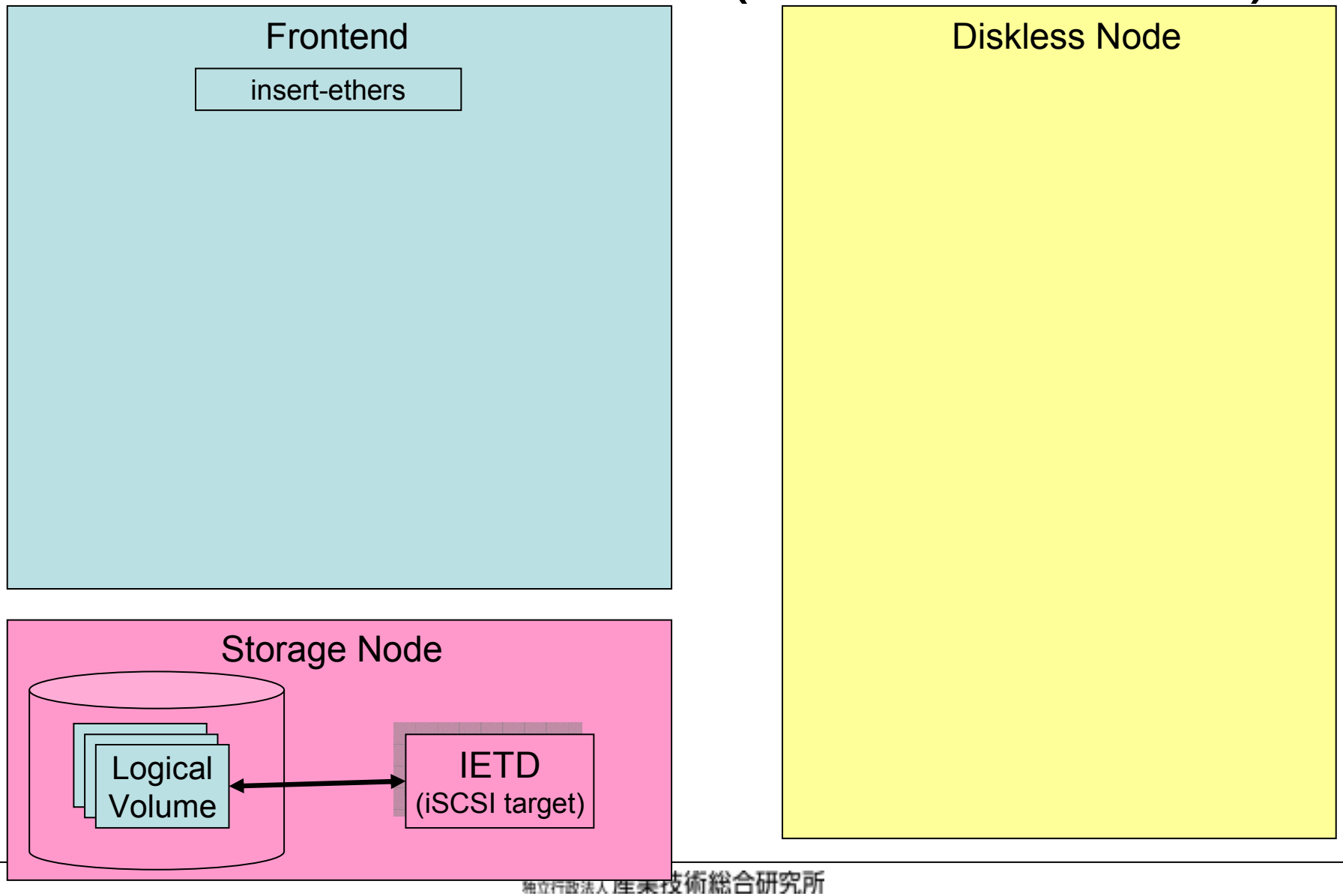
ディスクレス化機構の実装 (続)

- insert-ethersプラグイン
  - ストレージノード上にディスクレスシステム用のストレージのアロケート
  - iSCSIターゲットの設定
- ディスクレスロール
  - プレ処理: iSCSIイニシエータの設定
  - ポスト処理: フロントエンドに用意したXMLRPCサービスの呼び出し
- XMLRPCサービス
  - インストール完了後に次回ディスクレスブート用のカーネルオプション、initrdの再生成

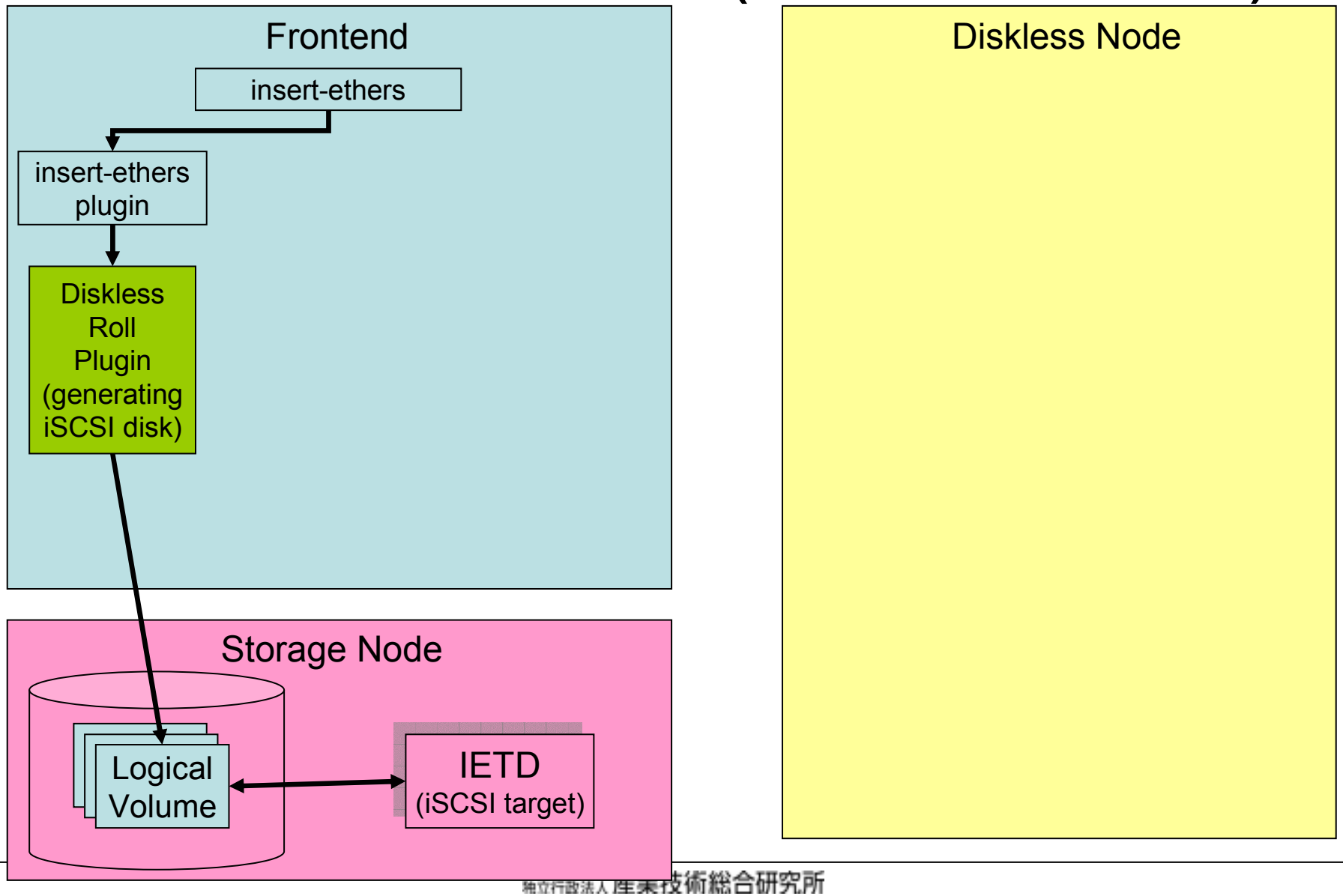
ディスクレス化機構(インストール時)



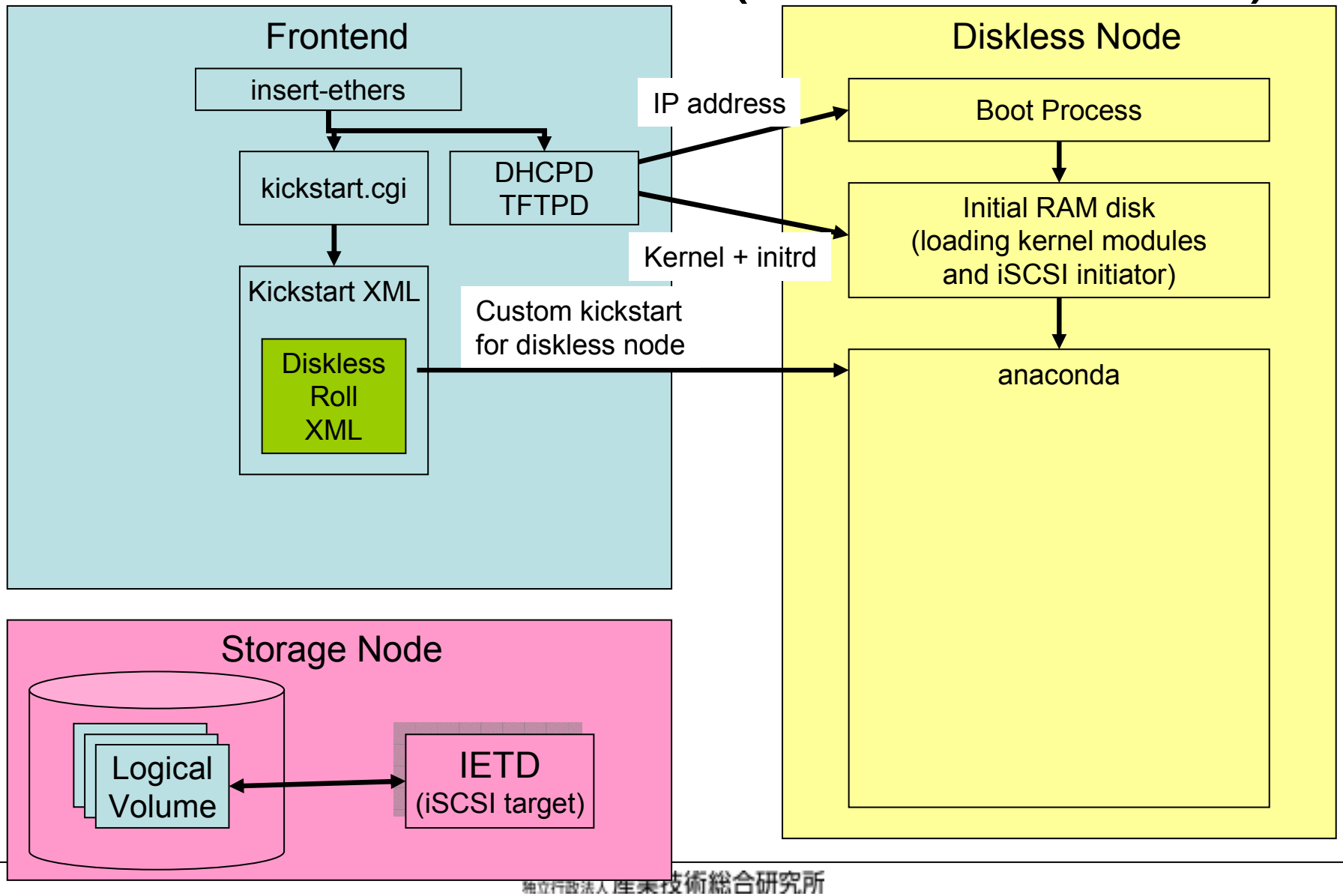
ディスクレス化機構(インストール時)



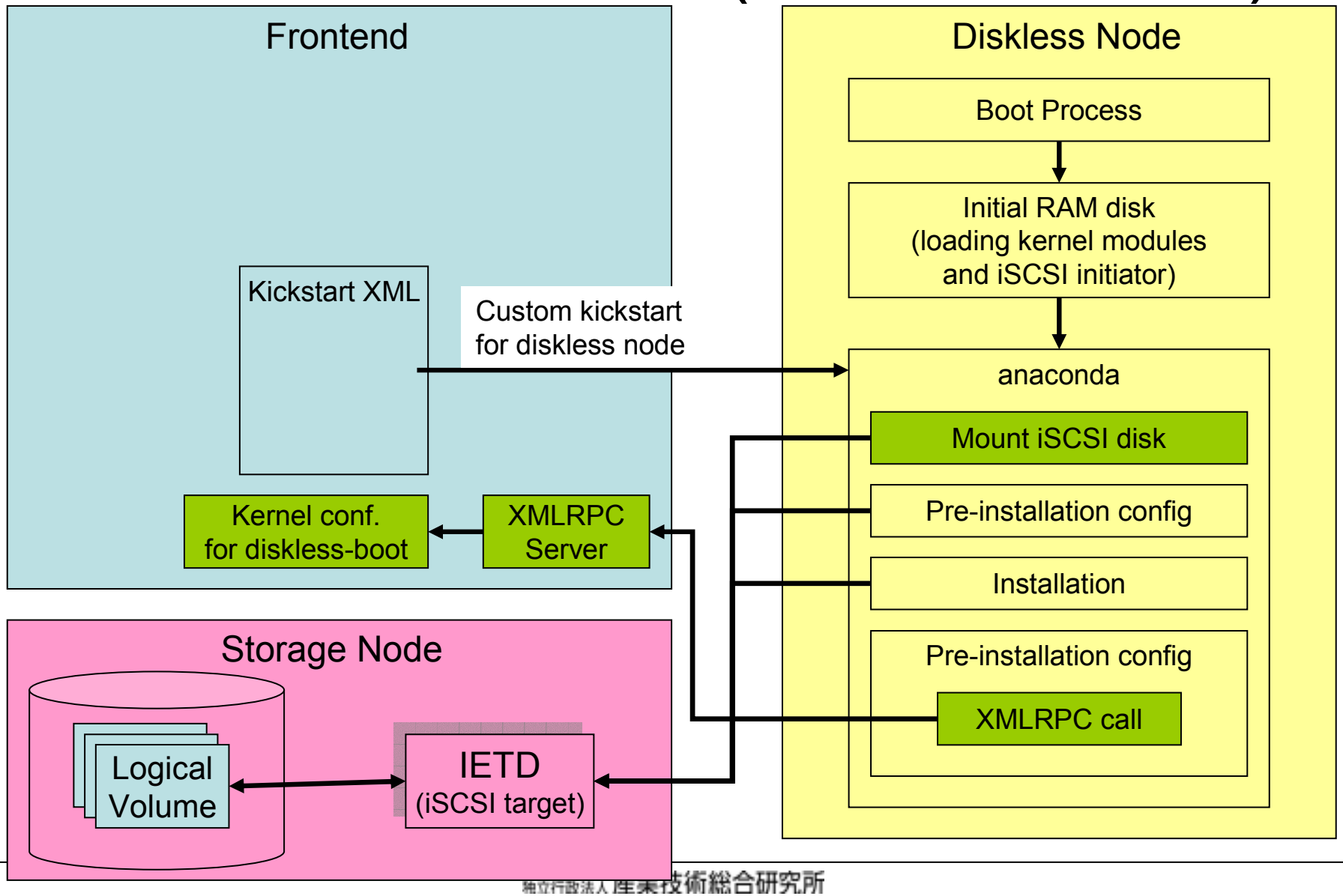
ディスクレス化機構(インストール時)



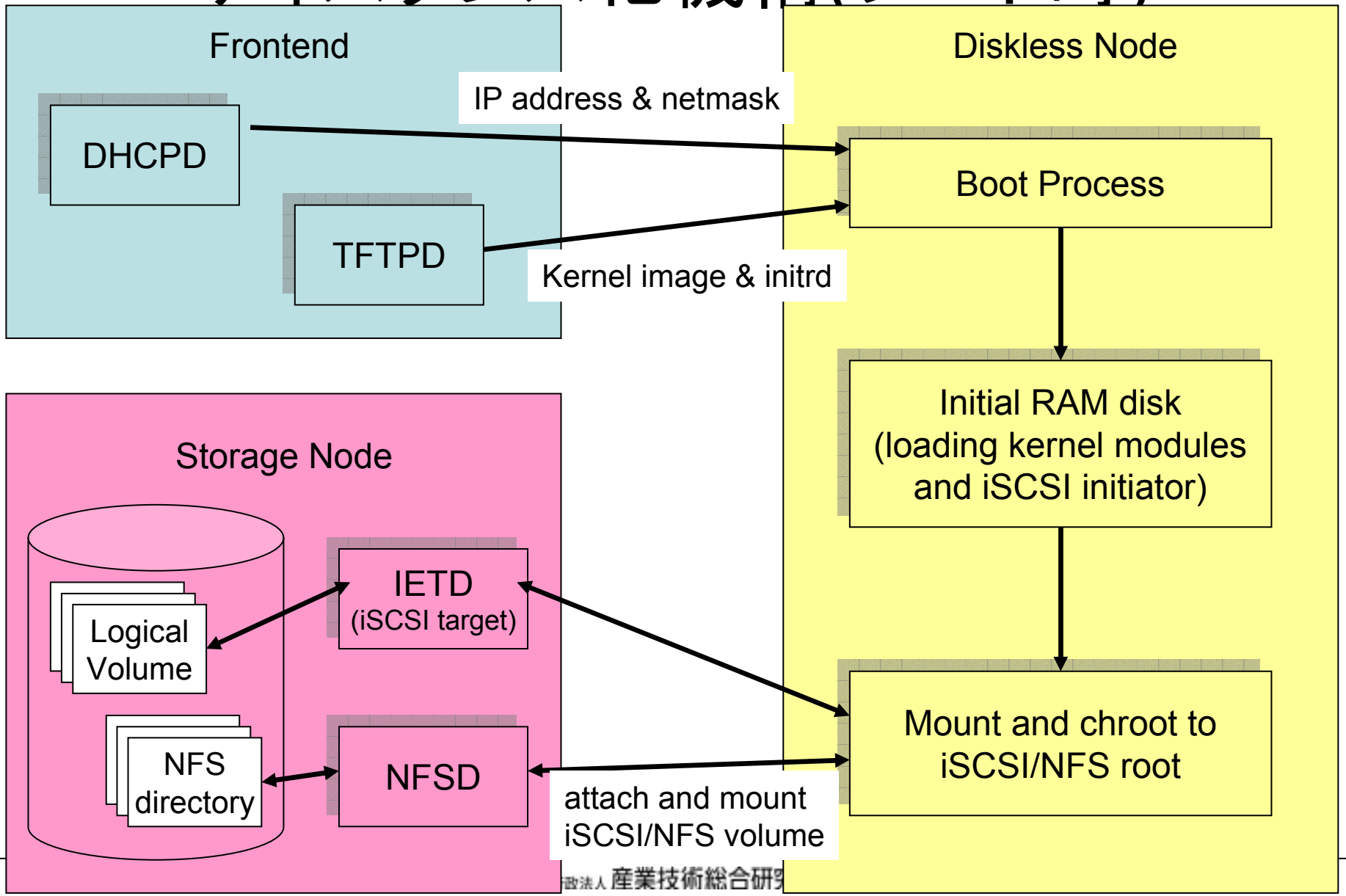
ディスクレス化機構(インストール時)



ディスクレス化機構(インストール時)

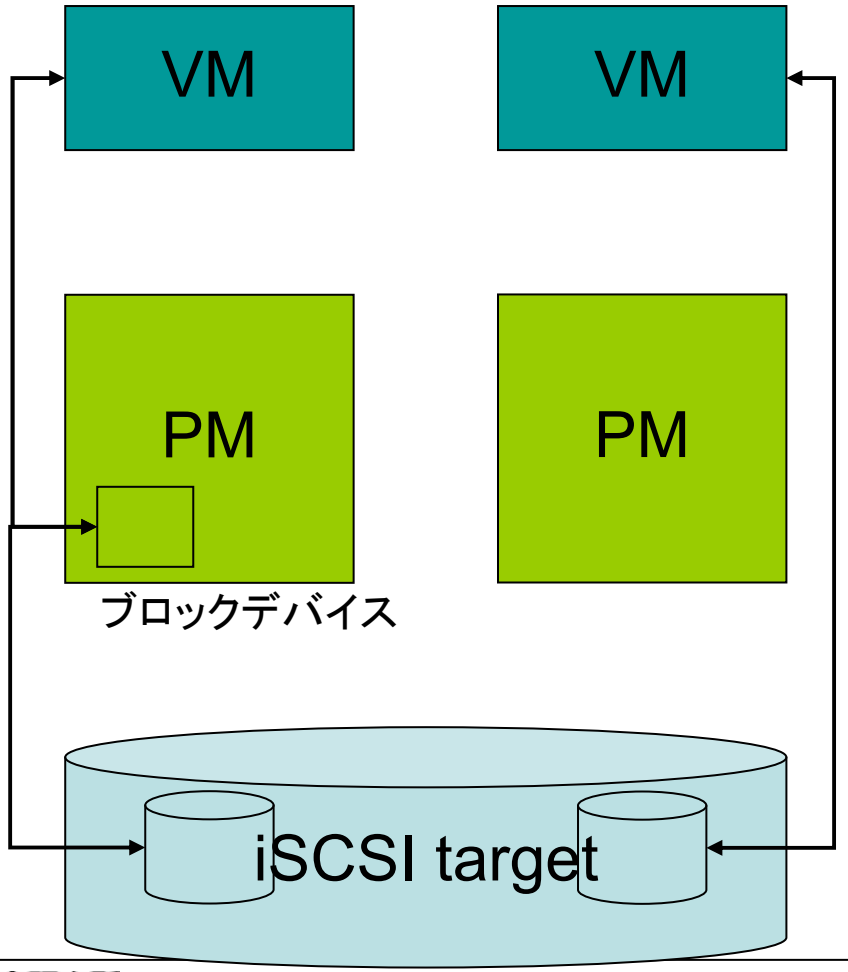


ディスクレス化機構(ブート時)



基本性能

- Core 2 Duo 1.86Ghz, 2GB
- 約500パッケージ、1GBを含むCentOS 5のインストール
- 双方とも約2000秒
- DHCPサーバやTFTPサーバとの通信オーバーヘッドよりインストール処理本体の分散の方が大きい



関連研究

- Diskless Roll
  - Rocksのディスクレスブート用ロール
  - NFS rootを強く仮定している
  - NFSサーバなどがフロントエンドに固定されてしまう
- Thin-OSCAR
  - OSCARクラスタインストーラでディスクレスブート機能を提供
  - initrdを手動で生成する必要があるなど未成熟な点がある

まとめ

- 仮想クラスタシステムのステートレス化のためのディスクレス化機構を実現
  - 物理マシン・仮想マシンが使用する、すべてのストレージを物理マシンから分離
  - ディスクレスブートの(再)活用
    = poor man's "boot from IP-SAN"
  - Rocks 5を基盤とした汎用的な実装
  - IP-SANの利用を前提とした場合、ディスクレス化によるオーバーヘッドは無視できる

今後の課題

- ストレージノードの「アプライアンス」化
- Grivonの次期バージョンとのマージ
- ライブマイグレーションの検討
- 実用的な規模での性能評価