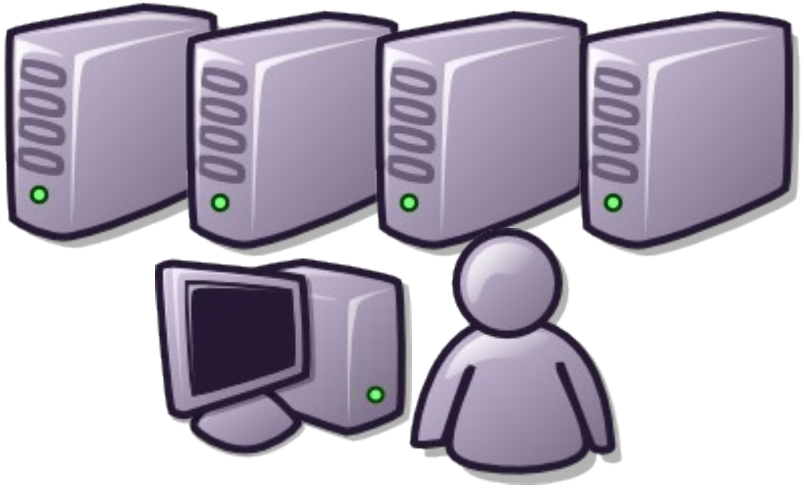
NPACI Rocks Cluster Toolkit 管理運用ガイド

住商情報システム
HPCソリューション技術部
飯坂剛一

目次

- PART1: インストール
- PART2: アーキテクチャ
- PART3: カスタマイズ
- PART4: 運用
- PART5: ジョブ管理

PART1: インストール



ソフトウェア構造とロール

並列計算プログラム / WebFarm / 開発Farm / Grid

メッセージパッシング / 通信レイヤー
MPI, MPD, Socket

ジョブ投入システム

ソフトウェア管理

クラスタ状態管理/モニタ

RHEL / CentOS /Sicentific etc...

Linux Kernel-2.6

ドライバソフトウェア
SRPMS
OFED, Myrinet, e1000

base

hpc

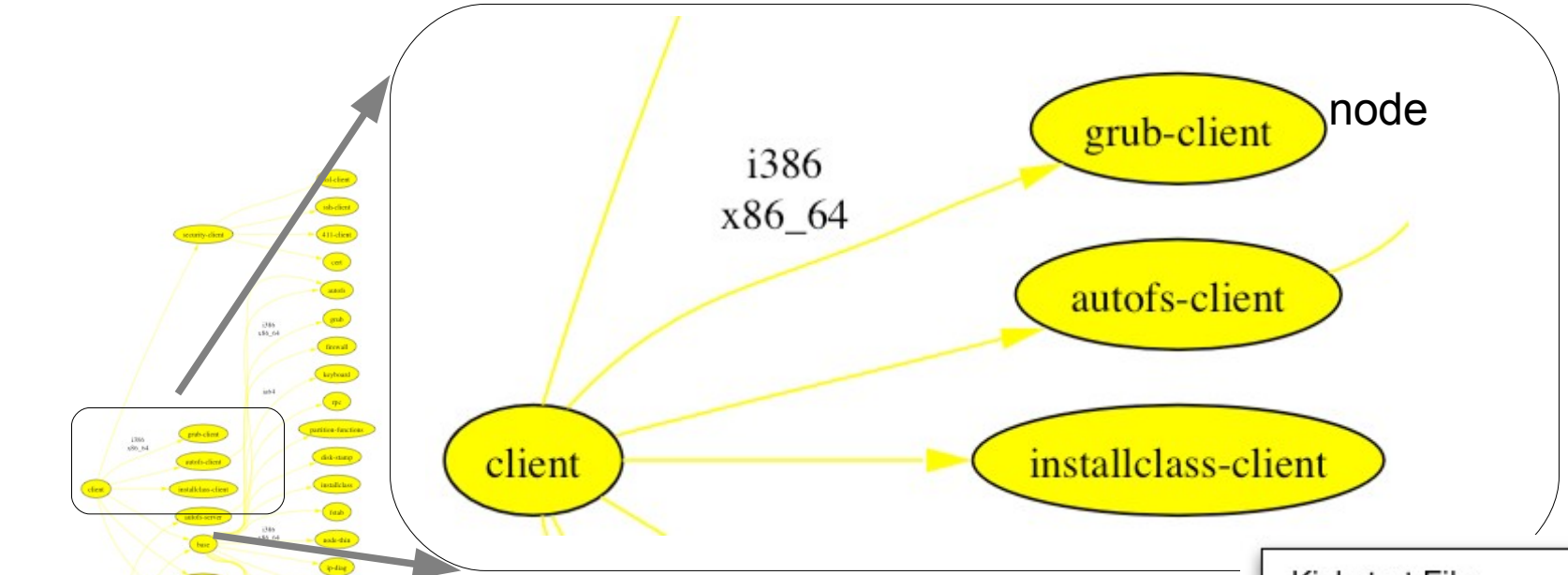
torque

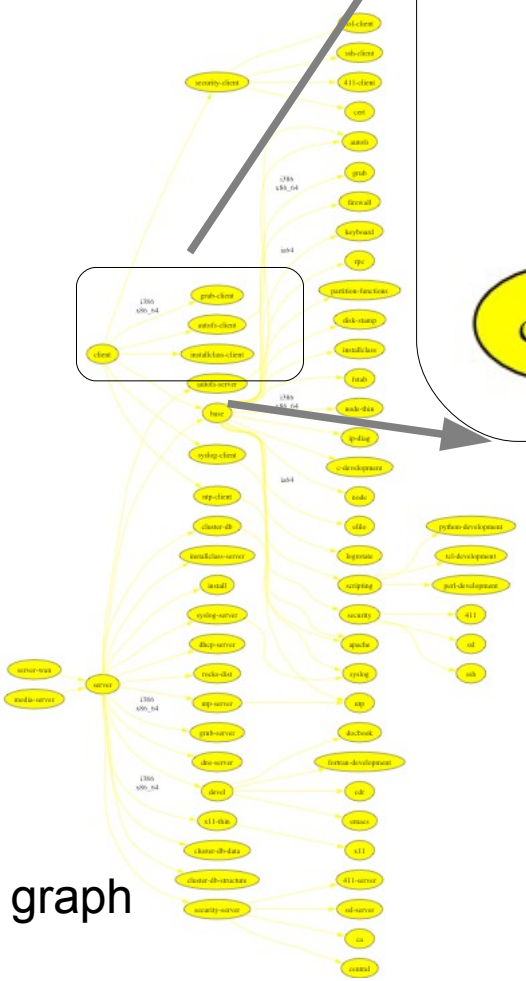
appli

Rollについて

- RPMパッケージとXMLKickstartをまとめたISOイメージ (CD or DVD)
- XML Kickstart
  - ダイナミックkickstarファイル
  - ノードファイル (node)
    - パッケージと手続きを記述
    - マクロ利用や外部プログラムの呼び出しOK
  - グラフ (graph)
    - インストール時の選択と順序を記述
  - ReferenceGuideに詳細情報
    - http://www.rocksclusters.org/rocksdocumentation/reference-guide/4.3/rocks-referenceguide-4.3.pdf

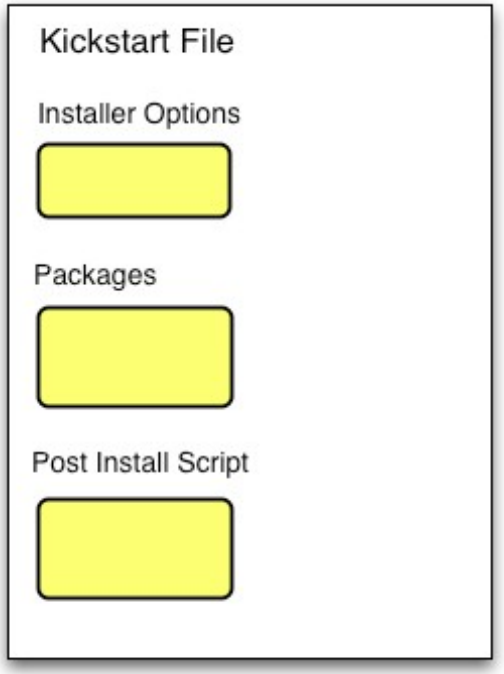
XML Kickstart





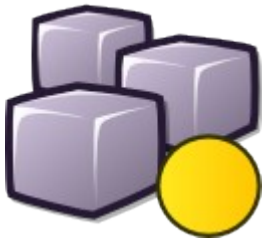
graph

Kickstartファイルを
ダイナミックに生成

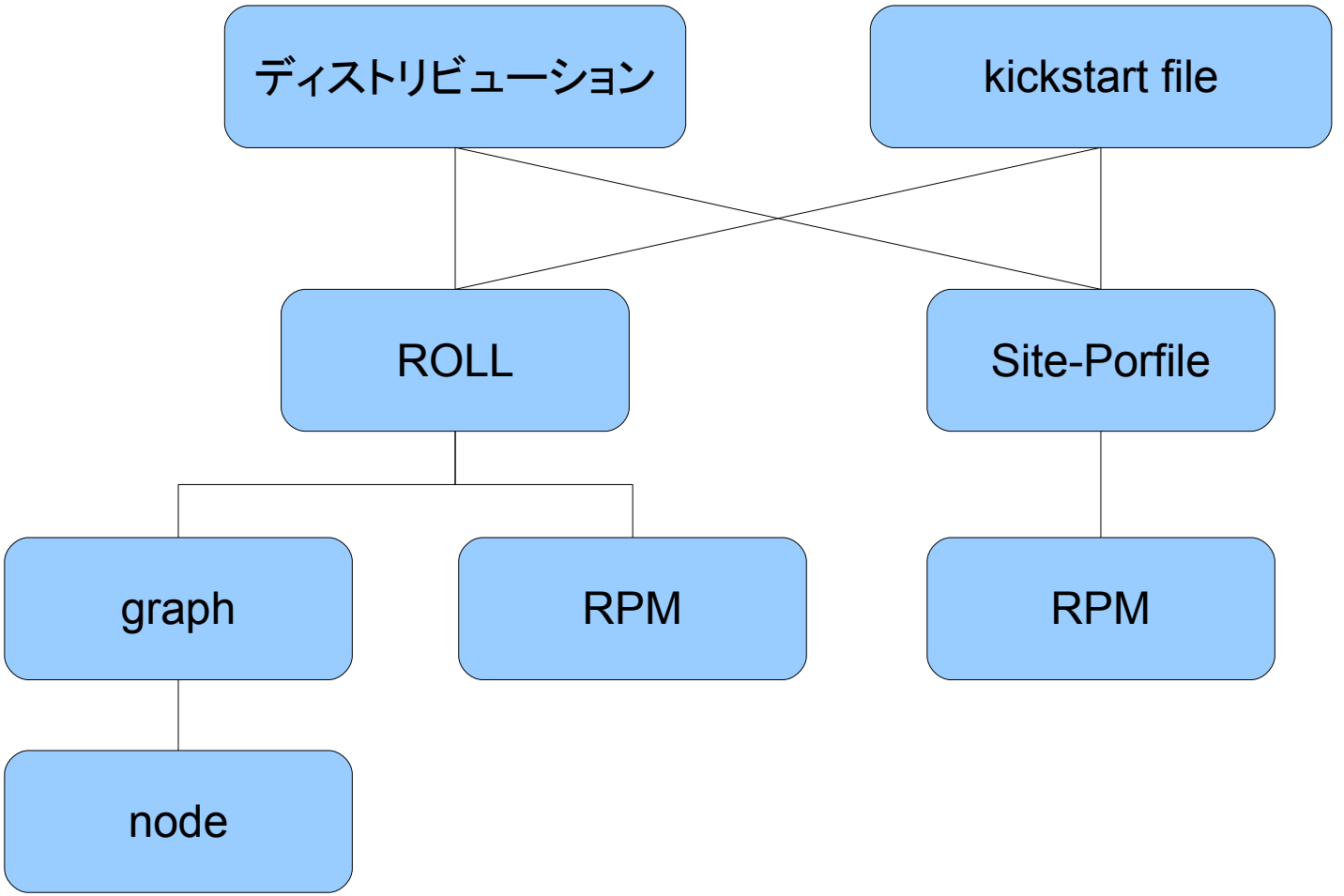


ディストリビューション

- 子ノードをインストールするためのRPMが格納されたディレクトリツリー
- kickstartファイルに記述されたRPMがフロントエンドノードからダウンロードされる
- rocks-distコマンドで作成される
  - /home/install/rocks-dist/lan/<ARCH>/RPMS



包含関係

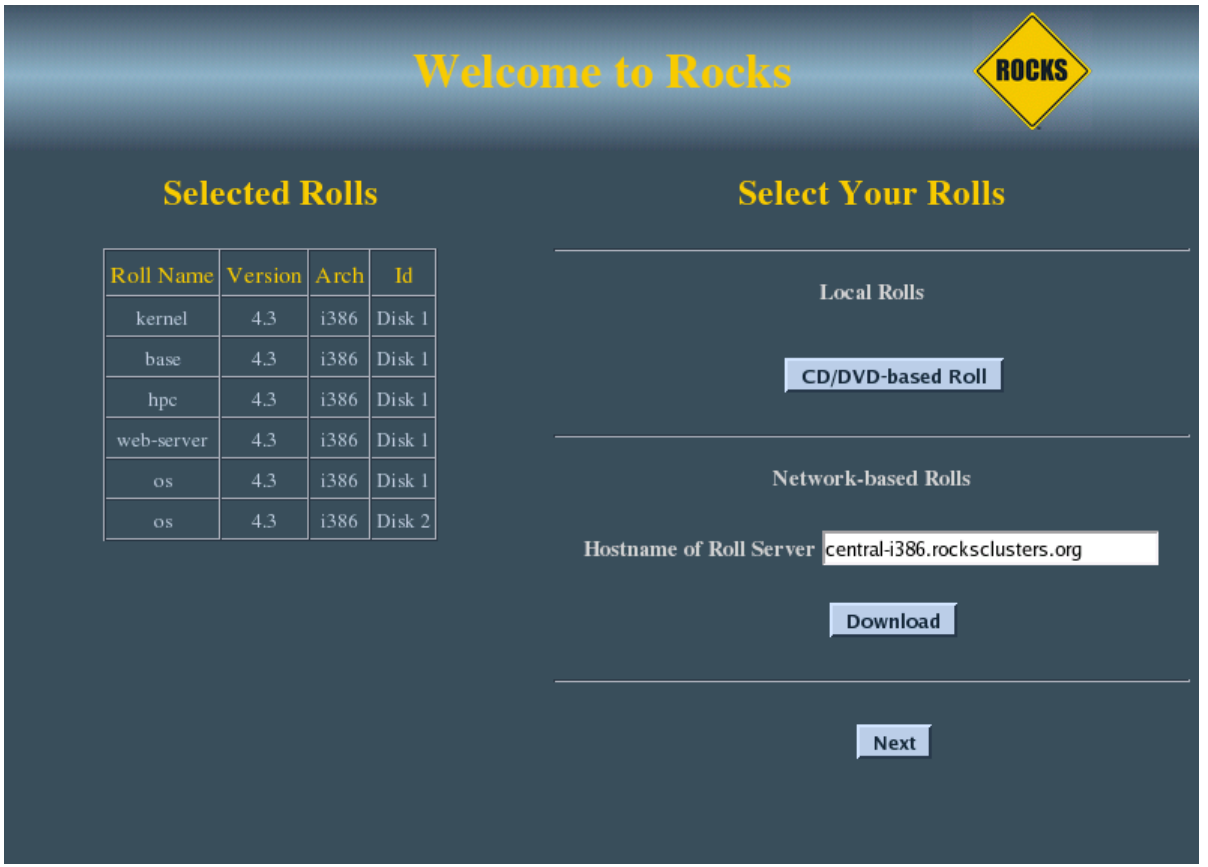

ディストリビューション
kickstart file
ROLL
Site-Porfile
graph
RPM
RPM
node

インストール

- 必須ロール
  - kernel, base, hpc, web-server, ganglia, OS
  - kernelロールがブートディスク

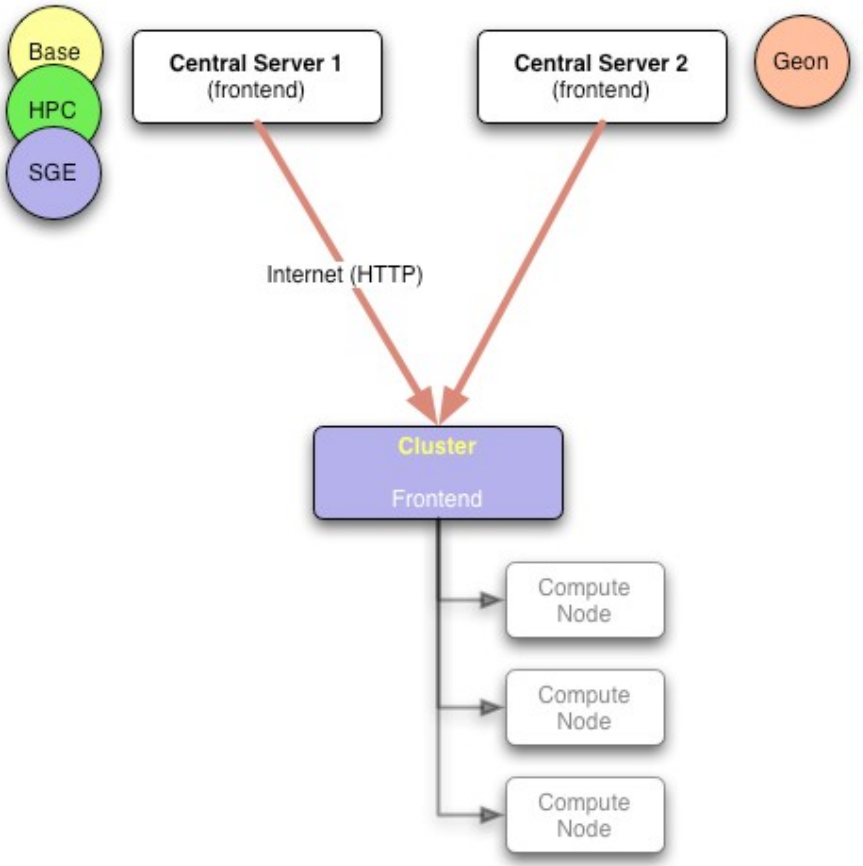
インストール

- オプションロール
  - OFED, GlusterFS, pfvfs2, sge, torque, viz...
  - セントラルサーバからのダウンロードもOK



セントラルサーバ

- フロントエンドノード自身
- Internet経由でもOK
- 横展開や成果共有が簡単
  - BootCDだけでOK
  - 拠点間で同じクラスタを保有

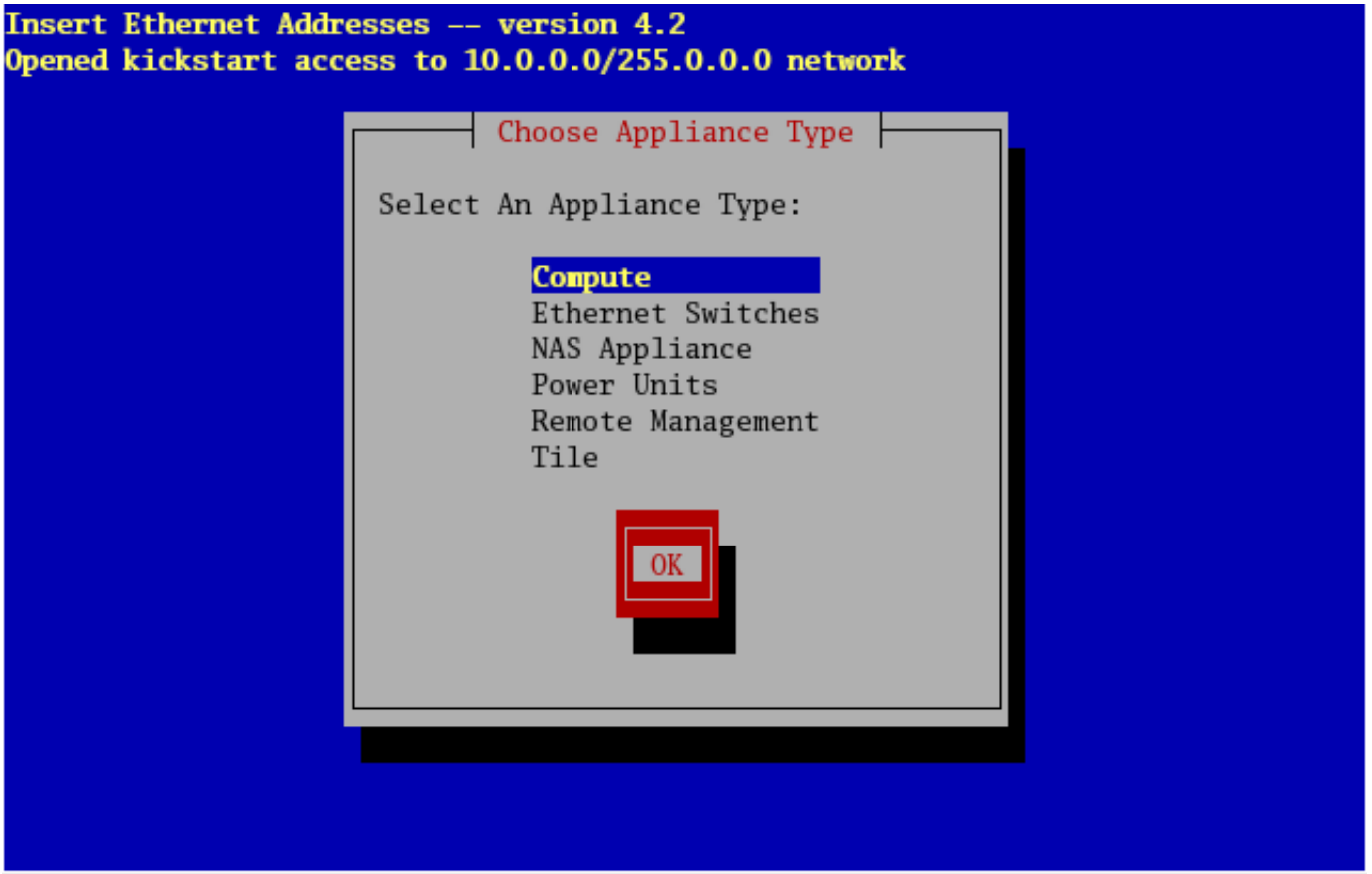


インストール時のTIPS

- KernelロールがBootCDになる
  - boot: linux
- レスキューモードでも起動できる
  - boot: linux rescue
- 16GB以上メモリが搭載されていれば mem=1024M のカーネルパラメタを与える
  - boot: linux mem=1024M
- このときキーボードはASCII配列なので留意のこと
- Kernelロールでブートさせて放置すると計算ノードをインストールするモードでインストーラが起動する

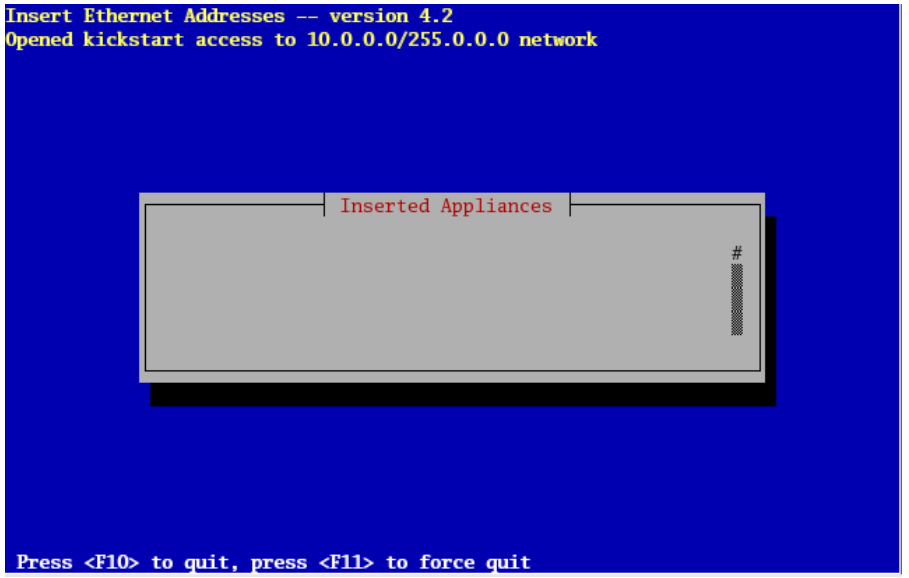
計算ノードのインストール

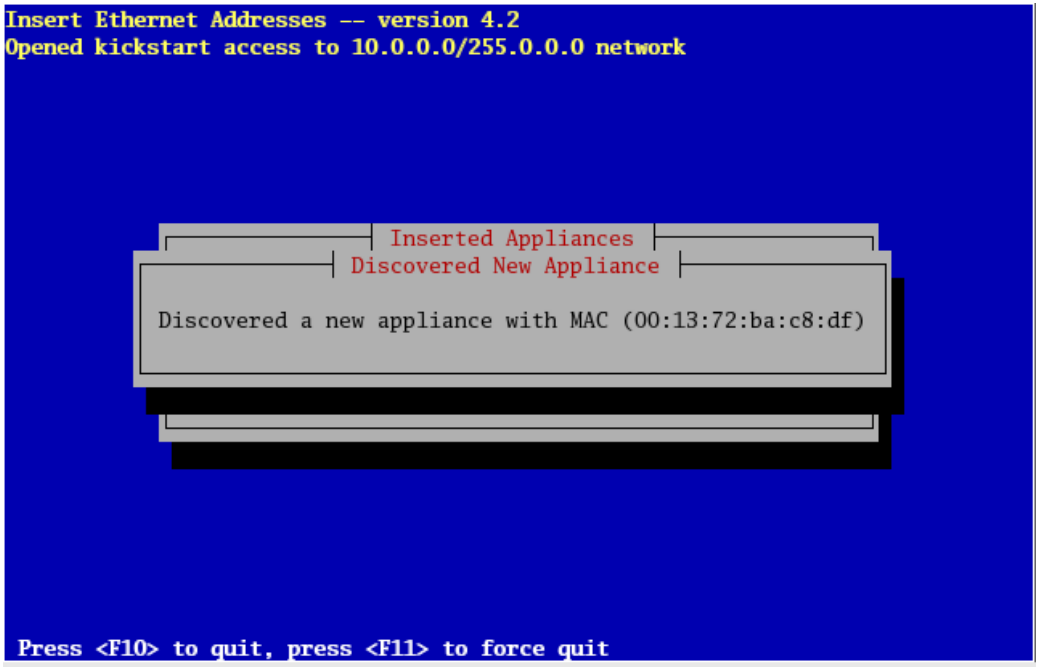
- フロントエンドノードで insert-ethers を実行
  - appliance を選択



計算ノードを電源ON

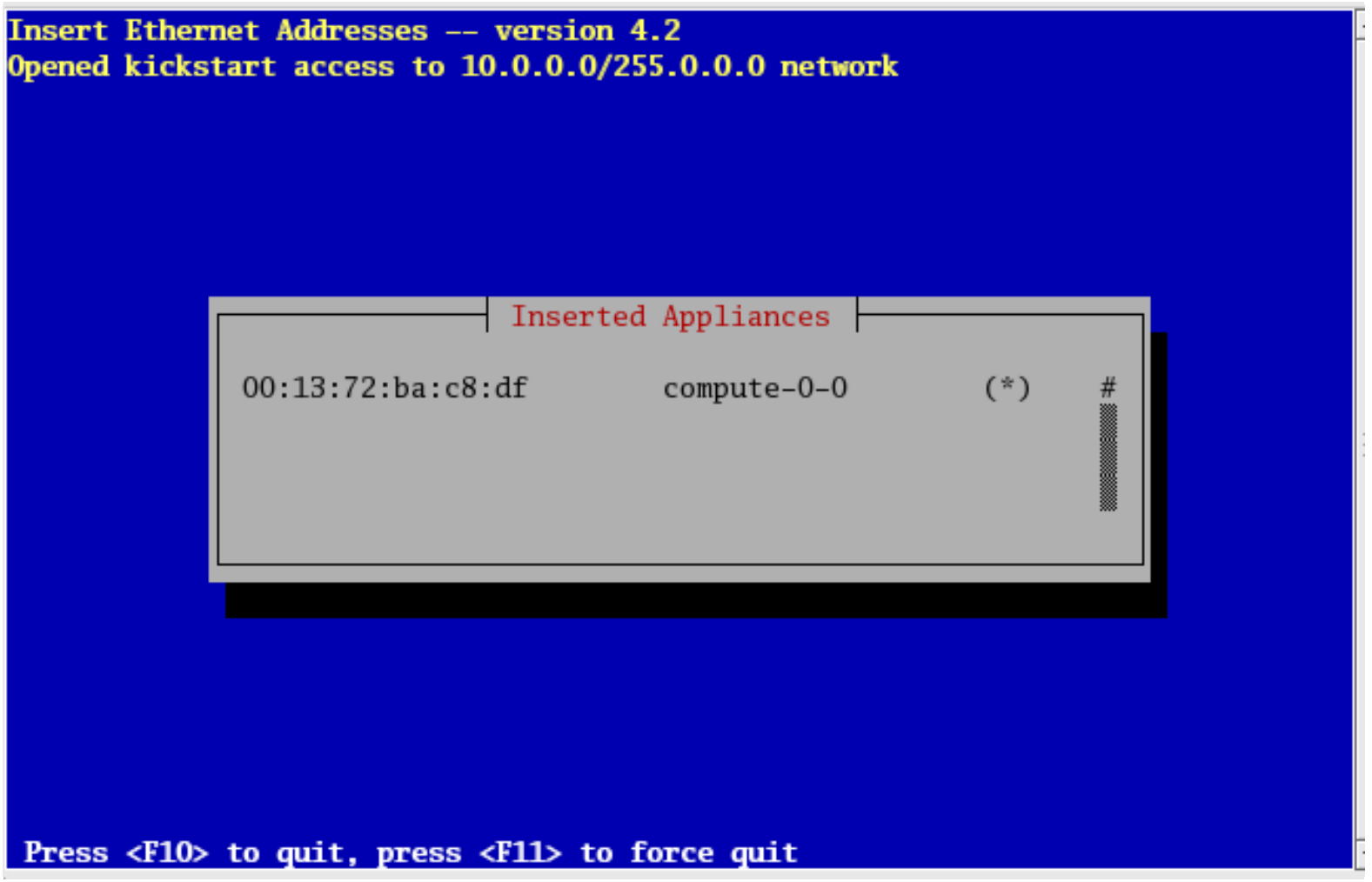
- MACアドレス自動収集
- インストーラダウンロード





インストーラの起動確認

- インストーラが起動すると (*) になる



計算ノードのインストール

- PXEブートによりインストーラをダウンロード
- インストーラがフロントエンドからkickstarfを取得
  - HTTPプロトコルを利用
- TIPS
  - インストール状況や不具合ではログを参照
    /etc/httpd/logs/{access_log,error_log}
  - httpd-devel が最後にダウンロードされるRPM
  - 411関連ファイルのダウンロードでブート終了
  - rocks-console ホスト名
    - 計算ノードへのNetwork KVM

HTTPの設定確認

- インストーラはHTTPプロトコルを使ってファイルを取得している
- アクセス制限などはapacheの設定にしたがう
  - /etc/httpd/conf.d
- rocksのデフォルトでは
  - /home/install/*/lan は計算ノードからアクセスできる

PART2: アーキテクチャ

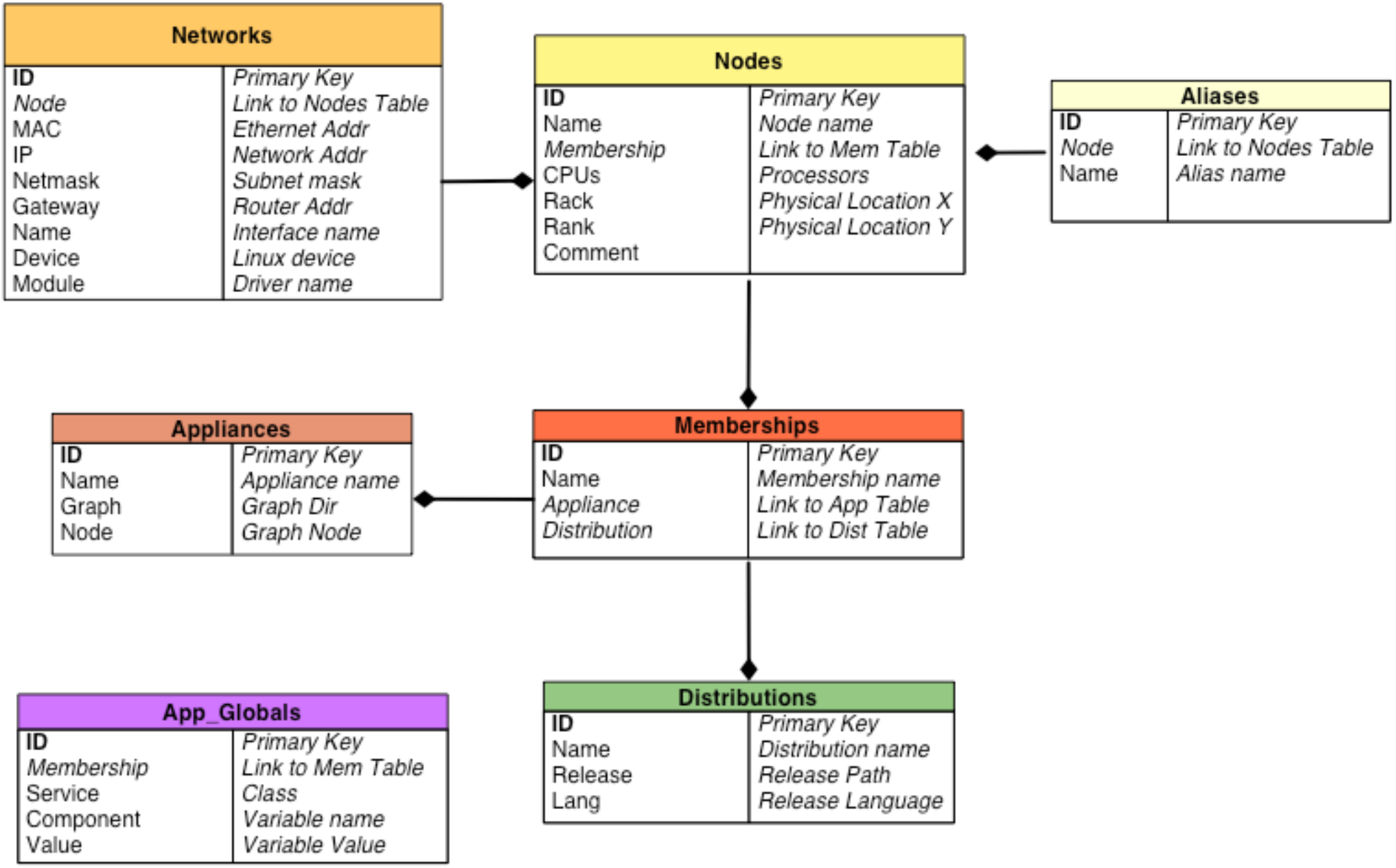
データベースによる一元管理

- NPACI Rocks ではクラスタシステムの各種情報をデータベースにより一元管理を行なっている
  - DBMS: MySQL
  - データベース: cluster
- アクセス
  - SQLによる方法： mysql -u apache cluster
  - dbreportコマンド: データの部分参照のみ
  - rocks コマンド： 修正もできる

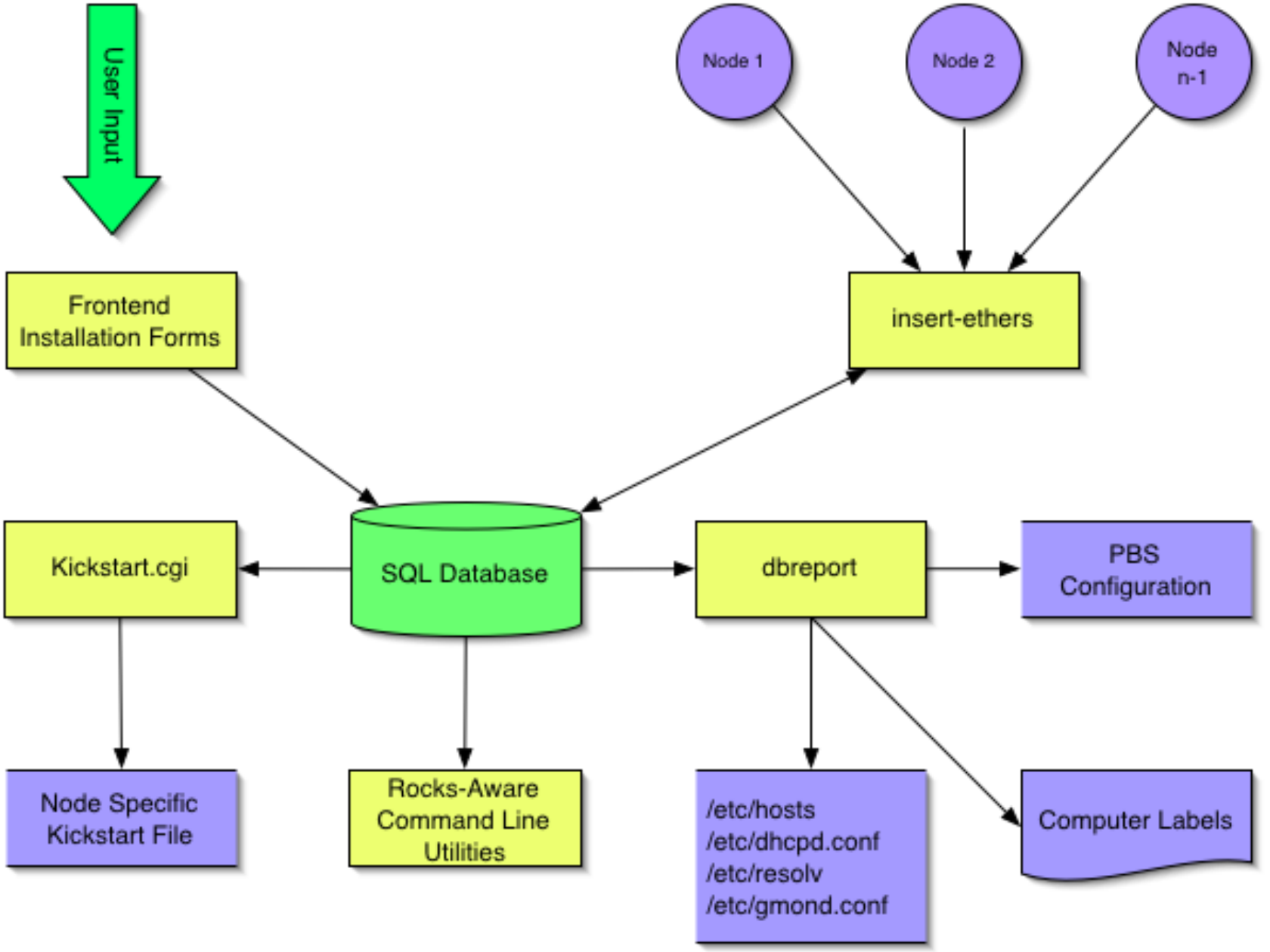
テーブル

- nodes
  - クラスタを構成するマシンのメンバシップを定義
- membership
  - メンバシップとapplianceをマッピング
- appliance
  - インストール時のgraphの起点ノードファイルを定義
- app_globals
  - グローバルデータ
- networks
  - クラスタを構成するマシンのネットワーク情報

データベーススキーム

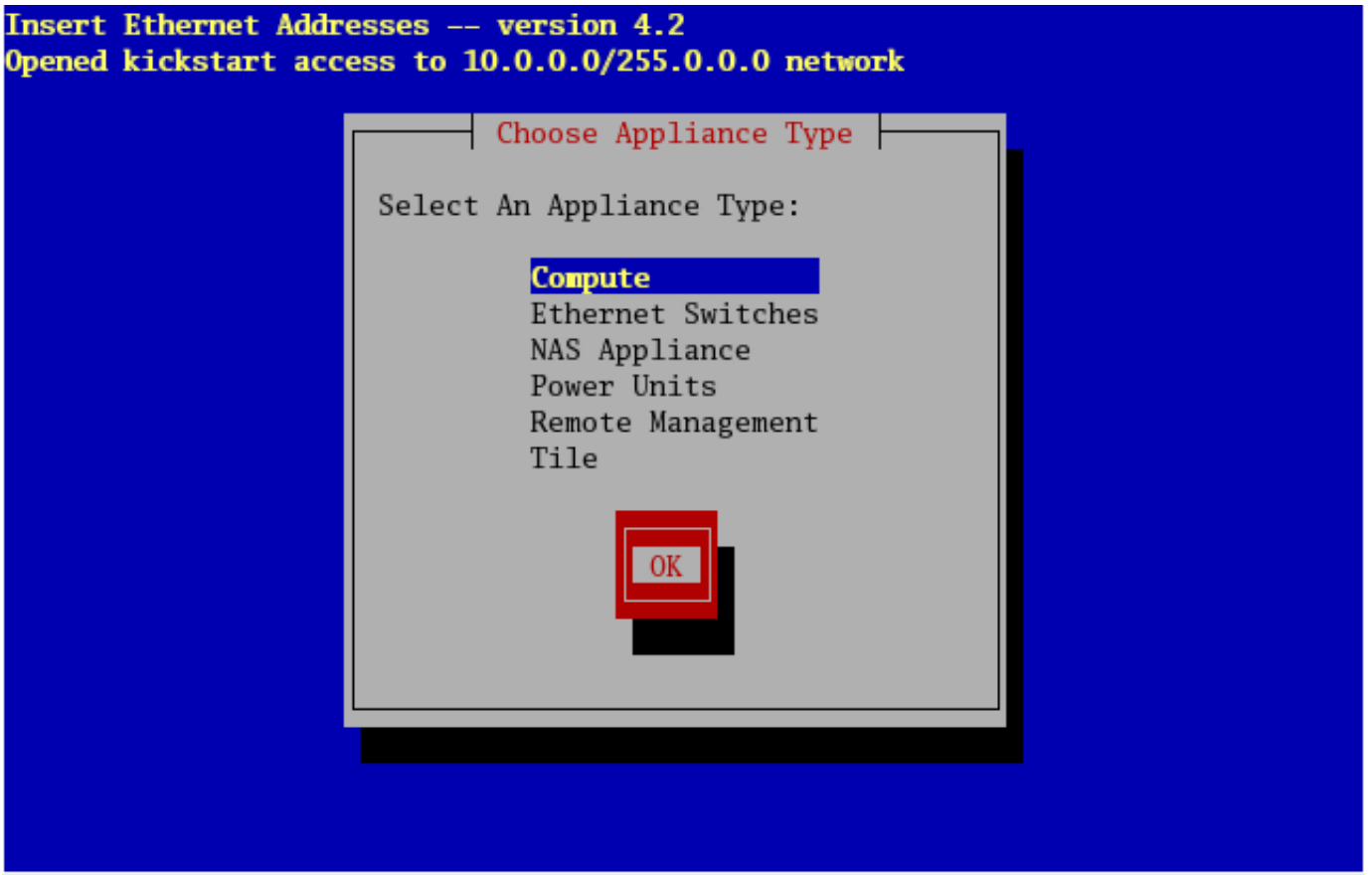

Networks
ID
Node
MAC
IP
Netmask
Gateway
Name
Device
Module
Primary Key
Link to Nodes Table
Ethernet Addr
Network Addr
Subnet mask
Router Addr
Interface name
Linux device
Driver name
Nodes
ID
Name
Membership
CPUs
Rack
Rank
Comment
Primary Key
Node name
Link to Mem Table
Processors
Physical Location X
Physical Location Y
Aliases
ID
Node
Name
Primary Key
Link to Nodes Table
Alias name
Appliances
ID
Name
Graph
Node
Primary Key
Appliance name
Graph Dir
Graph Node
Memberships
ID
Name
Appliance
Distribution
Primary Key
Membership name
Link to App Table
Link to Dist Table
App_Globals
ID
Membership
Service
Component
Value
Primary Key
Link to Mem Table
Class
Variable name
Variable Value
Distributions
ID
Name
Release
Lang
Primary Key
Distribution name
Release Path
Release Language

データフロー

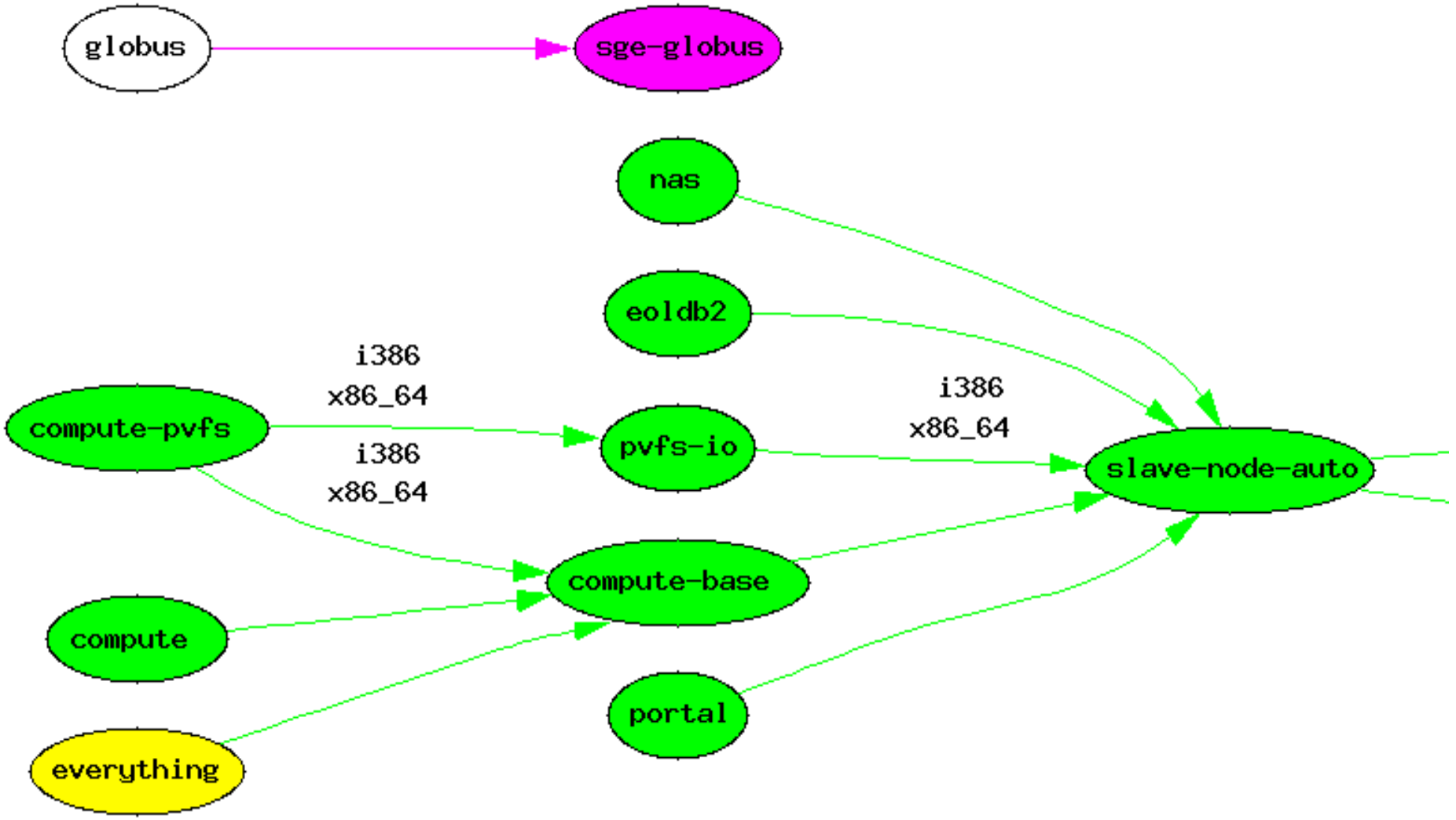

User Input
Node 1
Node 2
Node n-1
Frontend Installation Forms
insert-ethers
Kickstart.cgi
SQL Database
dbreport
PBS Configuration
Node Specific Kickstart File
Rocks-Aware Command Line Utilities
/etc/hosts
/etc/dhcpd.conf
/etc/resolv
/etc/gmond.conf
Computer Labels

graphの起点

- insert-ethers で選択する appliance で決定
  - applianceテーブルに起点となるノードファイルを定義

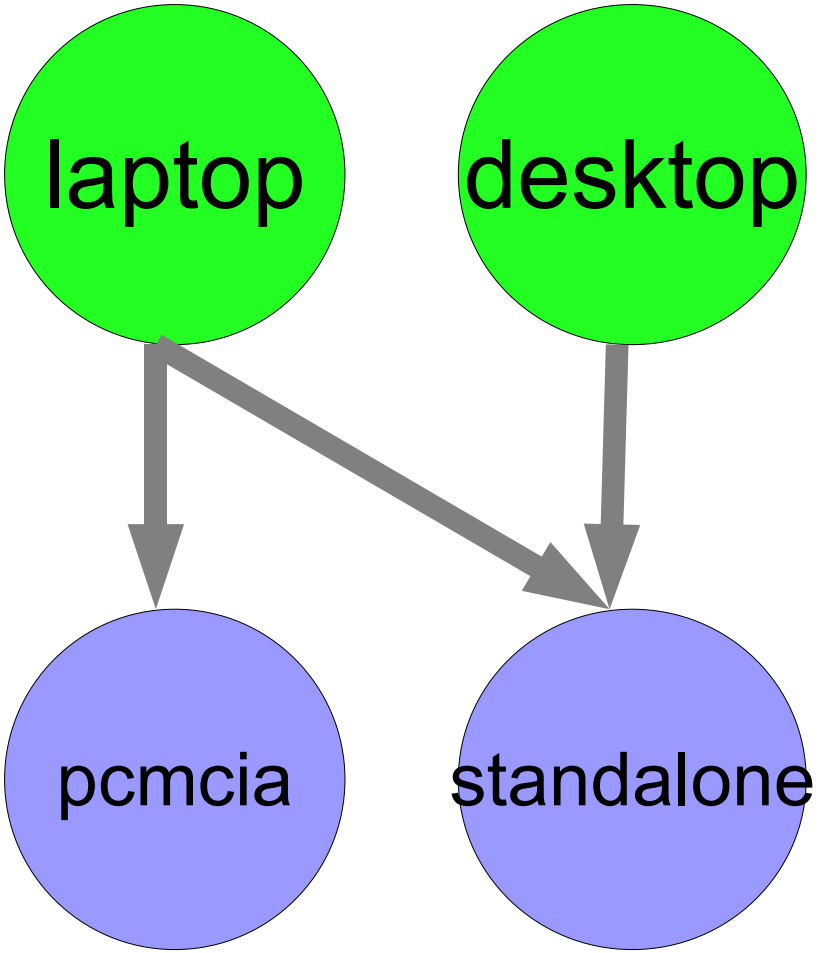


graphの起点


globus
sge-globus
nas
eoldb2
i386
x86_64
compute-pvfs
i386
x86_64
pvfs-io
i386
x86_64
slave-node-auto
compute-base
compute
portal
everything

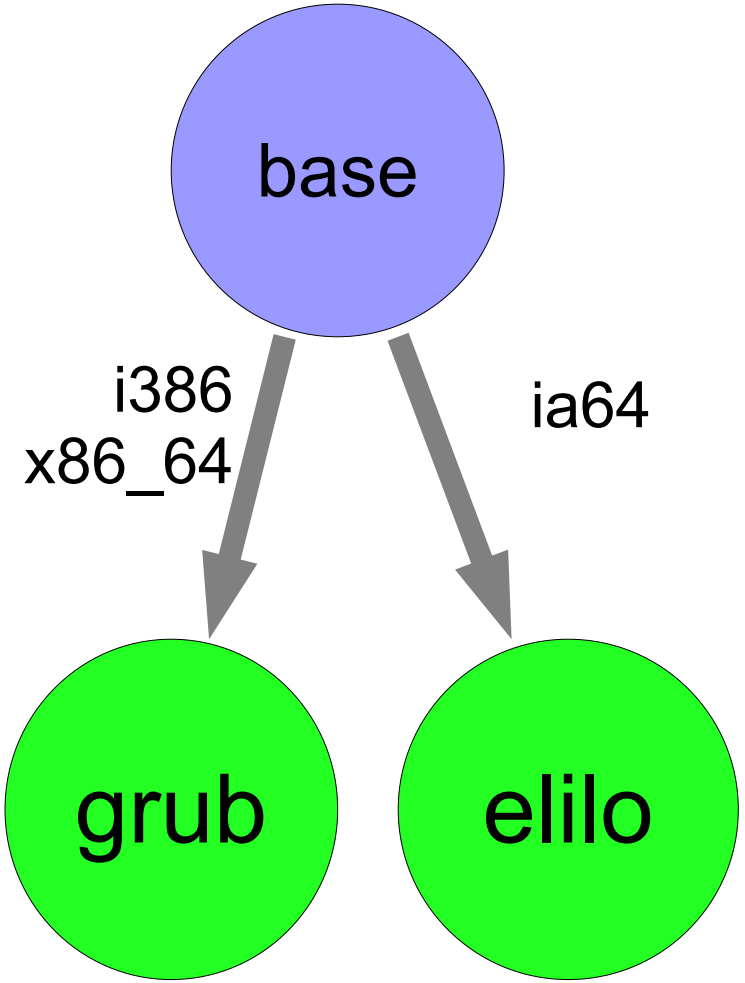
Applianceの補足説明

- Laptop / Desktop
  - Appliance
- Desktopが選択するノード
  - standalone
- Laptop が選択するノード
  - standalone
  - pcmcia
- ノードファイルの再利用

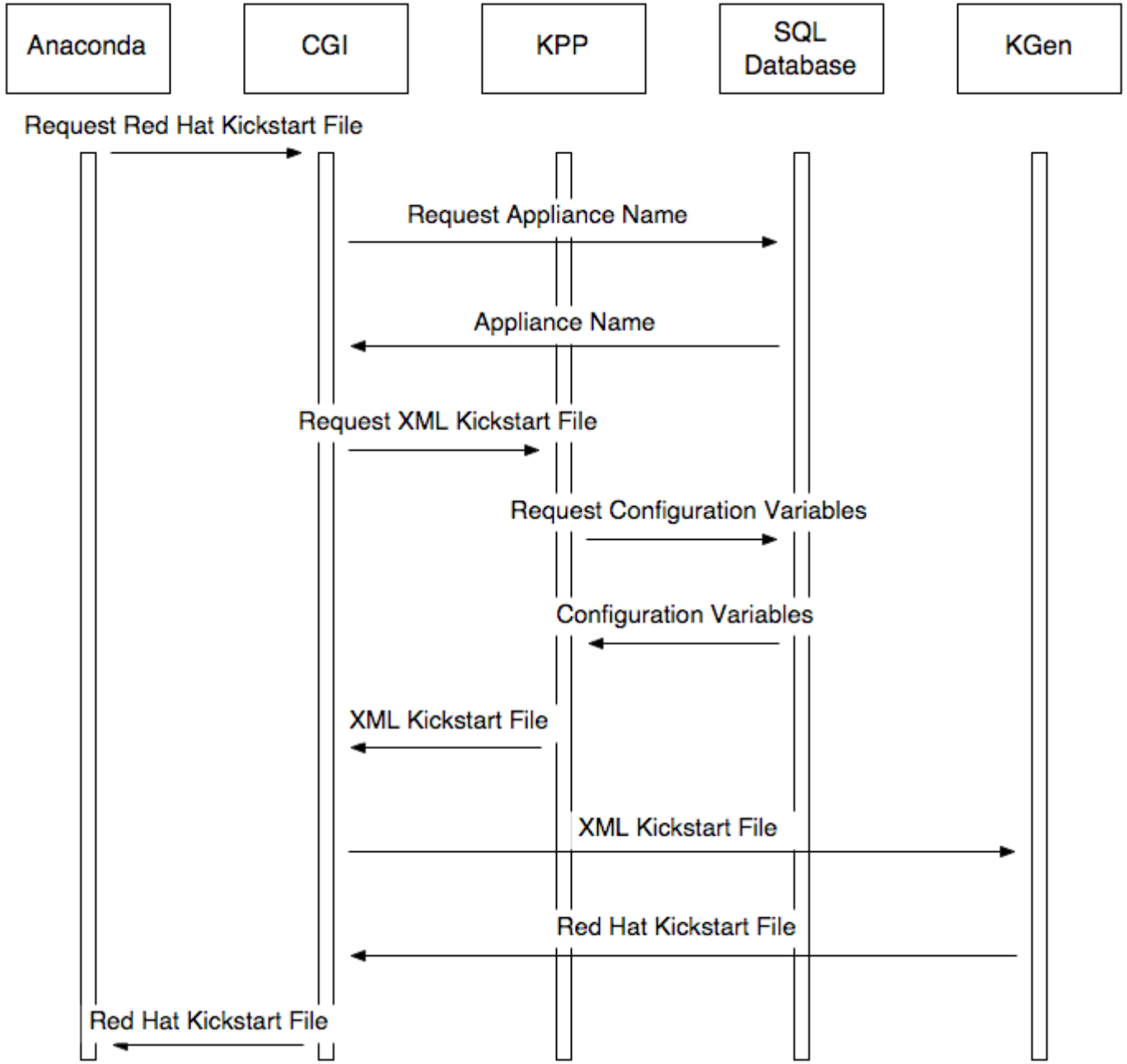


アーキテクチャ別の選定

- 条件付階層構造
  - edgeにアーキテクチャを記述
  - <edge arch=”x86_64”>
- もしi386なら
  - base はgrubを使用する
- もしia64なら
  - base はeliloを使用する
- 1つのGraphで異なるアーキテクチャをサポート
- ゴールデンイメージをコピーするクラスタではできないこと



インストール時のタイムライン



PART3: カスタマイズ

ロール管理

- インストールされているロールを一覧
  - rocks list roll
- ロールを追加する
  - rocks add roll clean=1 ロール名
- ロールを有効にする
  - rocks enable roll ロール名
- ロールを無効にする
  - rocks disable roll ロール名

ディストリビューションの構築

- 計算ノードがダウンロードするディレクトリを構築
  - /home/install/rocks-dist/lan/<ARCH>/RPMS
  - シンボリックリンク
- rocksオリジナルでは rocks-dist コマンド
  - cd /home/install
  - rocks-dist dist
- RPMの検索PATH
  - rocks-dist paths
    - /usr/src/redhar/RPMS
    - /home/intall/rolls
    - /home/install/contrib

追加ロールをインストール

- rocks-dist dist でディストリビューションを構築
- kroll コマンドでインストールスクリプトを生成
  - kroll　ロール名　>　postinstall.sh
  - sh　postinstall.sh
- 計算ノードへのインストールは再インストールでOK
  - cd /home/install
  - make reinstall

計算ノードへのパッケージ追加

- サイトプロファイルに記述
  - /home/install/site-profiles/4.3/{graph,nodes}
  - ひな型は skelton.xml
  - cp skelton.xml extend-compute.xml
- ファイル名での挙動
  - extend-ノード名
    - ノード名の処理が終わったあとに処理
  - replace-ノード名
    - ノード名の処理を置き換える

計算ノードへのパッケージ追加

- packageタグにRPM名を記述
  - <package> zsh </package>
  - RPMファイルが長くて名前がはっきりしないとき
    - rpm -qpi rpmファイル

packageタグ

- パッケージをインストールさせない
  - <package disable=1> RPM名 </package>
- メタグループの指定
  - <package type="meta"> GNOME </package>

OSとのパッケージ競合回避

- バージョン、リリースを参照し新しいものを使用
  - 独自に作成したパッケージで同じものがOSにも
    あるときに問題になる
- 自作パッケージを強制させたいとき
  - 方法1：自作した方のリリース番号を上げる
  - 方法2：forceディレクトリに置く
    - /home/install/rocks-dist/lan/<ARCH>force
    - カーネルのダウングレード時などで使用することがある

ロールでの競合回避

- ロールを自作したときのパッケージ競合回避
  - packageタグではなくpostタグでRPMをダウンロード
    - wget を使用
    - IPはインストール時に決定されるためマクロを使用

```
<post>
BASEURL=http://<var name="Kickstart_PrivateKickstartHost"/>
DIR=<varname="Kickstart_PrivateKickstartBasedir"/>/\
rocks-dist/lan/x86_64/RedHat/RPMS

wget  -o /dev/null  ${BASEURL}/${DIR}/YOU_RPM_FILE.rpm
rpn -Uvh –force –nodesp YOU_RPM_FILE.rpm
rm -f YOU_RPM_FILE.rpm
</post>
```

fileタグ

- インストーラ中でファイルを作成、追加時に使用
  - <file name=”ファイル名” perms=”755”>
  - 終端の</file>までがファイルになる
- アペンドする場合は
  - <file name=”ファイル名” mode=”append”>
- コマンドの実行結果をファイルの内容にする
  - <file name=”ファイル名” eval=”プログラム”>
- cat > ファイル名 <<'_EOF_' と等価
  - シェル環境変数 ${変数名} が使えないことに注意

nodeファイル記述での注意点

- XMLファイルであるため使用できない文字がある
  - & → &
  - < → <
  - > → >
- shellスクリプトとして記述できるがリダイレクトや条件文でハマりやすい
- XMLスクリプトに変換するのが面倒であればwgetでダウンロードして実行という方法もある

/etc/rc.d/rocksconfig.d

- このディレクトリに置いたpre*, post*のスクリプトはSYSV RCスクリプトの前後に起動される
  - RCスクリプトより前に実行： pre-*.sh
  - RCスクリプトの後に実行: post-*.sh

SRPMのビルド＆インストール

- ブート時にSRPMをビルドしてインストールする
  - 対象ディレクトリ:/opt/rocks/SRPMS
  - ビルドしてインストール
  - SRPMは削除される
- カーネルモジュールなどで便利
  - NICドライバ, fuse...
- ブート時にスワップ設定した後に処理される
  - ビルド時の標準出力、標準エラー出力への出力はファイルに格納されるため、多数のSRPMや大きいSRPMなどでは「止まってブートしない」と勘違いしやすい

ロール管理での利点

- ロールを有効/無効にできるために、Updateの適用有無を簡単に行なえる
- updateのロールバックが簡単に行なえる
- ロールはISOイメージで作成されるため、バージョン管理が単純
  - ロールを開発する労力に十分に見合う運用コスト削減

フロントエンドノードのパーティション

- インストール時に”manual” で設定できる
  - /, /var, swap, /export は必須
- デフォルトのサイズ
  - /　　16GB
  - /var　　4GB
  - swap　1GB
  - /state/parition1　残り全部
    - “auto”で設定した場合
    - /export は シンボリックリンクとして設定される

計算ノードのパーティション設定

- データベースで設定する方法
  - サイズの変更のみ
  - マウントポイントはデフォルトのまま
  - データベースを変更後、ディストリビューションを再構築
  - 計算ノードは再インストールだけでOK

```
# rocks list var | grep Partsize
Kickstart PartsizeRoot              8000
Kickstart PartsizeSwap              1000
Kickstart PartsizeVar               4000

# rocks set var Kickstart PartsizeRoot 16384
# rocks set var Kickstart PartsizeSwap 2048
# rocks set var Kickstart PartsizeVar  8192
```

計算ノードのパーティション設定

- サイトプロファイルで設定する方法
  - auto-partition.xml をコピーして replace-auto-partition.xml を作成
  - マウントポイントを変更できる
  - ソフトウェアRAIDなど複雑な設定もできる
  - データベースを変更後、ディストリビューションを再構築
  - 計算ノードは再インストールだけでOK

```
<main>
   <part> /     --size 16384  --ondisk  sda </part>
   <part> /var  --size 8192   --ondisk  sda </part>
   <part> swap  --size 2048   --ondisk  sda </part>
   <part> /work --size  1 -grow -ondisk sda </part>
</main>
```

パーティション設定の留意点

- データベースのpartitions テーブルの各ノードのパーティション設定情報が保存される。
  この情報があれば、まずこちらが優先される。
- compjute-0-1のパーティション情報を削除
  - rocks-partition –delete –noname=”compute-0-1”
- compjute-0-*のパーティション情報を削除
  - rocks-partition –delete –noname=”compute-0- %”
  - % がワイルドカード (SQLに渡しているから）

設定変更の確認

- ディストリビューションの再構築
- kickstartファイルを生成させて内容確認
  - dbreport kickstart compute-0-0 > /tmp/ks
- エラーがでたり内容がおかしいとき
  - cd /home/install/rocks-dist/lan/<ARCH>/build
  - kpp > /tmp/junk
    - 記述がおかしいXMLファイルがあればエラー表示される
    - サイトプロファイルを修正して再挑戦

計算ノードの再インストール

- shoot-nodeコマンド
  - 指定したノードだけ再インストールを行なう
  - 全ノードを行なう前に確認しておくために実行
  - shoot-node compute-0-0
- 全ノードの再インストール
  - cd /home/install ; make reinstall
  - 各ノードで次のことを実行している
    - rm -f /.rocks-release
    - /boot/kickstart/cluster-kickstart

PXEブート時の挙動

- PXEブート時の挙動はデータベースで管理
- 再インストール
  - rocks set host pxeboot ホスト名 action=install
- memtestを実行
  - rocks set host pxeboot ホスト名 action=memtest
- OSで起動
  - rocks set host pxeboot ホスト名 action=os
- 全ノードにactionを指示
  - rocks-pxeaction [-v] [os|install|memtest]

計算ノードのエイリアスホスト名

- add-alias コマンドを使用
  - エイリアス名の追加/削除/一覧


```
# add-aliases --add -nodename=cluster --alias="frontend"
# add-aliases --list

Node              Aliases
-----------------------------------------------
cluster    frontend
```

ノード追加時に処理をしたい

- insert-ethers のプラグインを記述する
  - ノード追加時の呼び出される
  - /opt/rocks/var/plugins/insertethers
  - サンプルは sample.py
  - pythonで記述する
    - コマンドの実行だけなら os.system(“コマンド”) を挿入

TORQUEでの補足説明

- ラック毎にグルーピングを行なっている
  - アトリビュート cg1 〜 cg4 を各ノードにアサイン
- /opt/torque/server_priv/nodes.attr で定義
  - TORQUEが読むのは nodes
  - nodes はinsert-erthersのプラグインで作成される
    - nodes.attr を読み込んで nodes を作成している

PART4: 運用

一時的なパッケージの追加

- yum によるパッケージ管理
  - yum install パッケージ名
  - 依存パッケージも合わせてインストール
- yum レポジトリのメンテナンス
  - cd /home/install/distro
  - make repo
- yumがダウンロード時に参照するディレクトリ
  - /home/install/share/lan/CentOS
    - /home/install/distro/CentOS/4.6 へのシンボリックリンク
    - 計算ノードからもダウンロードできるようにしている

ユーザ管理

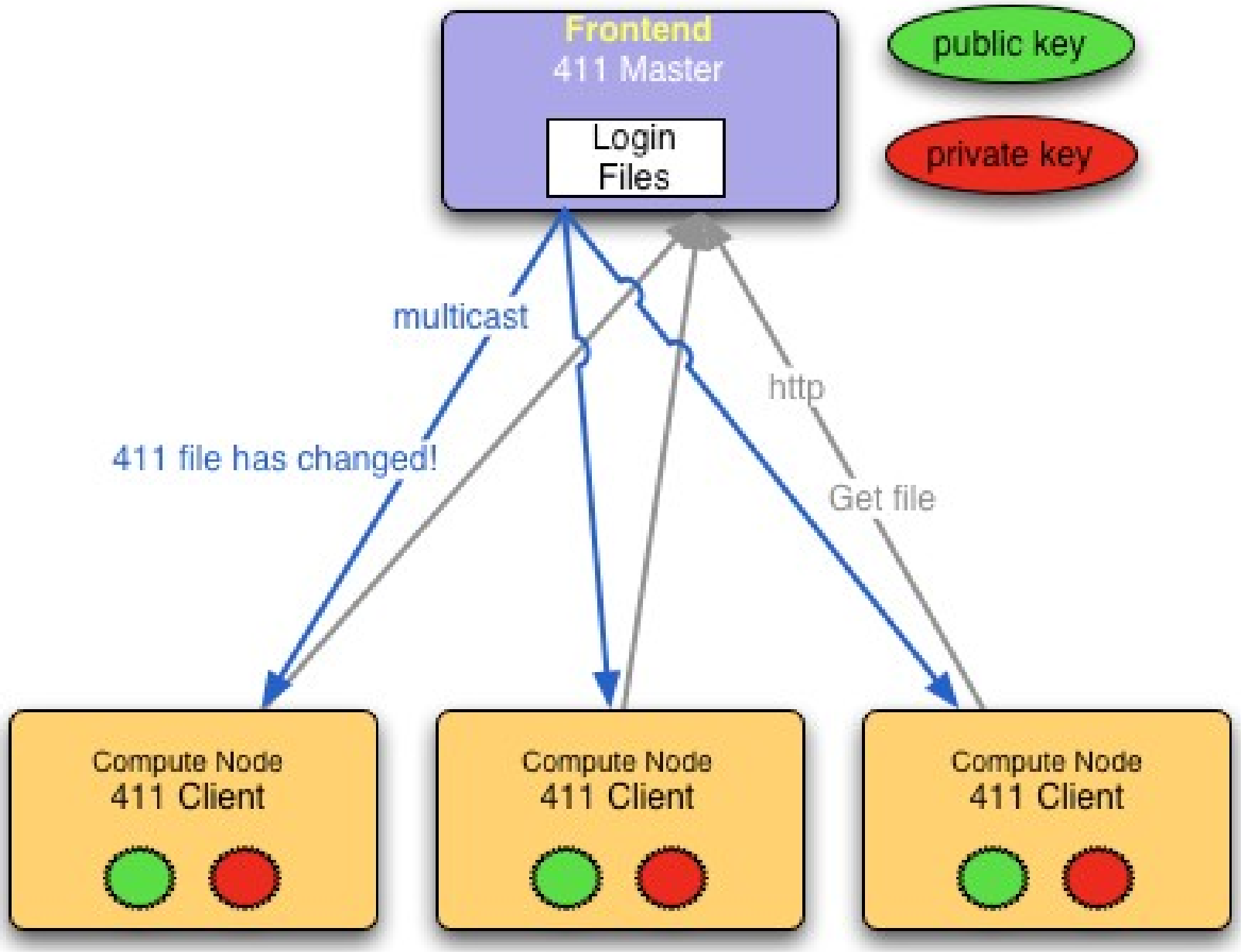
- useradd　ユーザ名
- passwd　ユーザ名
- rocks sync users
  - /etc/auto.home にエントリ追加
  - 計算ノードへユーザ情報を配布
- 外部ファイルサーバをホームディレクトリにしたいときなどは /etc/auto.home を修正して
  再度 rocks sync users を実行

/etc/hostsへエントリ追加

- /etc/hosts.local に記述する
  - /etc/hostsは自動生成されるため、直接かけない
- /etc/hosts を生成
  - rocks sync config
  - dbreport hosts > /etc/hosts

どちらかでOK

411


Frontend
411 Master
Login
Files
public key
private key
multicast
411 file has changed!
http
Get file
Compute Node
411 Client
Compute Node
411 Client
Compute Node
411 Client

411とは

- 411 Secure Information System
  - 名前の由来はUSの番号問い合わせの電話番号から
- 登録したファイルを配布する
- ファイルは暗号化される
- マスターサーバはマルチキャストを発行
- クライアントはHTTPプロトコルを使いマスターサーバからファイルを取得し、複号してからコピー
- 1000ノード超クラスタでもほぼ同時に配布可能

411の仕様上での留意点

- ファイルの先頭部分にメタ情報が付加される
  - ファイルパス、モードなどがコメントとして追加
  - 全てのファイルタイプのコメントを知ってはいない
- シェルスクリプトを配布するとNG
  - #!/bin/bash などがファイルの先頭でなくなるため

411の操作

- ファイルを411へ登録
  - 411put /full/path/to/file
    - 絶対パスを指示する
- 登録されているファイルを一覧
  - 411get –list
- 登録されているファイルをすべて取得
  - 411get –all
- ファイルを指定して取得
  - 411get /full/path/to/file

更新されたファイルの配布

- /var/411/Files.mk
  - FILES に更新チェックを行なうファイルパスを指定
- make -C /var/411
  - /etc/411.d に登録されているファイルと、ファイルパスにあるファイルのタイムスタンプを比較して、更新があればマルチキャストを発行

各ノードへのコマンド発行

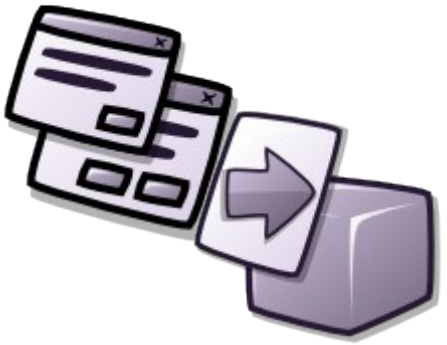
- cluster-fork
  - Rocksオリジナル
  - --sql=”SQL文”で対象を選択できる。(!?)
  - 途中で止められない
- tentakel
  - 通常はこちら
- ScatterCmd
  - 飯坂作：必要に迫られた機能あり。単純

ノード障害時の対応

- マザーボードが交換になったとき
  - MACアドレスの情報はデータベースに保存されている
  - マザーボードが交換されるとMACアドレスが変わる
  - rocksコマンドで変更する
    - rocks set host interface mac cluster eth0 MACアドレス
    - rocks set host interface mac cluster eth1 MACアドレス
  - rocks-replace-macコマンドでも変更できる
    - 会話型にも処理できる

scsbase

PART5: ジョブ管理



TORQUE

- OpenPBSのいろいろな不具合に対処
- 可読性のよいログ出力
  - OpenPBSのエラーコードをメッセージに変換している
- スケジューラ
  - 高機能スケジューラ maui を使用

TORQUEの起動/停止

- サーバの起動/停止
  - ブート時に自動起動される
  - /etc/init.d/pbs_server [start | stop]
  - /etc/init.d/maui [start | stop]
- monサーバの
  - ブート時に自動起動される
  - /etc/init.d/pbs_mon [start | stop]

ジョブでの前後処理

- prologue / eplogue というファイル名のスクリプトを用意することでジョブの前後に実行させることができる
- /opt/torque/mom_priv
  - 各計算ノードに配布する必要がある
  - 411では配布できないので留意

ジョブ投入

- qsub -q キュー名 myjob.sh
  - キュー名を指定して投入
- qsub -lnodes=3:ppn=2 -q キュー名 myjob.sh
  - -l リソース指定
  - ノードあたり2CPUを3ノードをリクエスト

ジョブ削除

- qdel -k ジョブID
  - 指定したジョブIDのジョブを削除
- qsig -s kill ジョブID
  - 指定したジョブIDのジョブにKILLシグナルを発行

ジョブをホールド

- qhold ジョブID
  - ジョブIDで指定したジョブをディスパッチさせない

checkjob: ジョブの状態

- /opt/maui/bin/checkjob

```
$ /opt/maui/bin/checkjob  395


checking job 395

State: Running
Creds:  user:scs  group:scs  class:q4g1s  qos:DEFAULT
WallTime: 00:00:00 of 00:10:00
SubmitTime: Fri Mar 28 14:56:43
  (Time Queued  Total: 00:00:01  Eligible: 00:00:01)

StartTime: Fri Mar 28 14:56:44
Total Tasks: 8
```

checkjob: ジョブの状態

- つづき

```
Req[0]  TaskCount: 8  Partition: DEFAULT
Network: [NONE]  Memory >= 0  Disk >= 0  Swap >= 0
Opsys: [NONE]  Arch: [NONE]  Features: [c4g1a]
Allocated Nodes:
[compute-0-0.local:2][compute-0-1.local:2][compute-0-2.local:2]
[compute-0-3.local:2]



IWD: [NONE]  Executable:  [NONE]
Bypass: 0  StartCount: 1
PartitionMask: [ALL]
Flags:       RESTARTABLE

Reservation '395' (00:00:00 -> 00:10:00  Duration: 00:10:00)
PE:  8.00  StartPriority:  1
```

showq: キューの状態

```
$  /opt/maui/bin/showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING
STARTTIME


     0 Active Jobs       0 of  256 Processors Active (0.00%)
                         0 of  128 Nodes Active      (0.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT
QUEUETIME


0 Idle Jobs

BLOCKED JOBS-----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT
QUEUETIME


Total Jobs: 0   Active Jobs: 0   Idle Jobs: 0   Blocked Jobs: 0
```

showstats: ジョブ稼働履歴

- 一般ユーザは使えない

```
# /opt/maui/bin/showstats

maui active for   17:23:51:26  stats initialized on Thu Jan  1
09:00:00

Eligible/Idle Jobs:                    0/0          (0.000%)
Active Jobs:                           0
Successful/Completed Jobs:           244/244        (100.000%)
Avg/Max QTime (Hours):              0.00/0.21
Avg/Max XFactor:                    0.53/1.27

Dedicated/Total ProcHours:       3589.39/238738.65 (1.503%)

Current Active/Total Procs:            0/256        (0.000%)

Avg WallClock Accuracy:              6.092%
Avg Job Proc Efficiency:          2913.147%
Est/Avg Backlog (Hours):               0.00/0.00
```