Vol.2018-HPC-163 No.18
2018/3/1

超大規模 HPC システムのインストール高速化の提案

小久保良輔[†1] 松山和広[†1] 三鴨利彰[†1] 住元真司[†1] 平井浩一[†1]

概要：これまで，HPC システムでは，各ベンダーから提供されたツールを利用して，システムインストールを実施してきた．近年，HPC のシステムソフトウェアではオープンソースソフトウェア(OSS)が広く使われはじめており，インストールに必要なツールについても OSS の利用が拡大してきている．そのため，ベンダーごとに異なる手法を用いるのではなく，汎用的な OSS を活用してインストールすることが HPC システム構築の主流となってきている．しかし，現状の OSS インストールツールは，大規模なシステムを想定した設計ではないため，超大規模なシステムに適用するにはいくつかの課題がある．本論文では，超大規模 HPC システムの構築における，現状の OSS インストールツールの課題と改善のアプローチを述べる．

1. はじめに

HPC のシステムソフトウェアではオープンソースソフトウェア(OSS)が広く使われている．近年，ソフトウェア単体の OSS 化だけではなく，OSS を活用した HPC ソフトウェアスタックとして OpenHPC [1] も提供され，HPC における OSS 利用は利便性も含めてますます拡大している．HPC システムのインストールツール(以下，HPC インストールツール)はこれまで各ベンダーから提供されたツールを利用して，システムインストールを実施してきた．今後はインストールにおいても，ベンダーごとに異なる手法を用いるのではなく，汎用的な OSS を利用することが主流になっていくと予想される．HPC システムでは，ノード台数を増やすことによりシステム性能を高めているが，現状の OSS の HPC インストールツールは，大規模なシステムを想定した設計になっていない．本論文では，超大規模 HPC システムの構築における，現状の OSS の HPC インストールツールに対する課題提起と，その一部について改善提案および，試作評価した．

2. HPC インストールツールの概要と要件

HPC インストールツールは，インストールサーバより，HPC システム内のノード群に対して，一括してリモート OS インストールを実施するツールである．これによって，システム管理者が 1 台ずつ手動でノードに OS のインストールや初期設定を実施するシステム構築作業を軽減することができる．本章では HPC インストールツールの一般的な機能と要件について述べる．

2.1 HPC インストールツールの一般的な機能

HPC インストールツールは大きく次に述べる 2 つの機能を有する．

(1) インストールサーバの自動設定

インストールサーバから HPC システム内のノード群に対してリモート OS インストールができるようにするために，インストールサーバ上で DHCP サービスや TFTP サービスなどインストールに必要な設定が存在する．これらの設定はインストールサーバ上で HPC システム内のノード(以下, インストール対象ノード)ごとに設定が必要である．HPC インストールツールはインストールサーバのこれらの必要な設定を自動で実施し，インストール環境を整備する機能を有する．

(2) リモート OS インストール

各インストール対象ノードはそれぞれホスト名や IP アドレス，ディスクパーティションなどノードごとに可変なノード情報を有する．HPC インストールツールはリモートから OS をインストールする際に，これらの可変なノード情報の設定を実施する．また，HPC システムでは，計算に利用する計算ノードや，システムを管理するための管理ノードなど異なる特性のノードが存在し，同じ特性のノードには同一のパッケージ群を適用するのが一般的である．

2.2 HPC インストールツールの要件

HPC インストールツールは，HPC システム構築時に，システム管理者が利用する．

HPC システムの構築は，一度に全ノードを構築するだけでなく，段階的にノードが増設されることや，クラスタ構成の変更が行われることがある．HPC インストールツールは，システム管理者が，短時間で容易に HPC システム内のノード群を構築，増設および，構成変更できるために，以下の要件が求められる．

(1) インストールサーバの自動設定に対する要件
インストールサーバからリモート OS インストールをするための環境整備が手早くできること

(2) リモート OS インストールに対する要件
HPC システム内のノード群を短時間でリモート OS インストールできること

3. HPC インストールツールの OSS 実装

本章では，2.2 節で示した HPC インストールツールの要件について，OSS における一般的な実現方法を述べる．

3.1 インストールサーバの自動設定

OSS の HPC インストールツールは一般的に，インスト

†1 富士通株式会社
FUJITSU LIMITED

ールサーバの設定を自動化する機能を有する．設定を自動化する手段として，CUI のコマンド(ノード登録コマンド)を利用して OSS の HPC インストールツールにノードを登録することで実現されている．システム管理者はインストール対象ノードの可変情報(ホスト名や IP アドレス，ディスクパーティションなど)をコマンドの引数としてノード登録コマンドを実行する．OSS の HPC インストールツールはノード登録コマンドで登録したこれらのノード情報をもとにインストールサーバの以下の設定を実施する．これにより，ネットワークインストールによるリモート OS インストールが可能となる．

(1) DHCP サーバの設定
ネットワークインストールにおいて，インストール対象ノードが利用するネットワーク情報(IP アドレスやブートローダ）など，DHCP サーバの設定を作成

(2) TFTP サーバ
ネットワークインストールにおいて，インストール対象ノードがインストールに必要な OS の情報(カーネルイメージ，initrd イメージなど)を取得できるよう，PXE 設定ファイルを作成

(3) ファイルサーバ
インストール対象ノードがインストール部材(適用するパッケージ群など)をダウンロードするためのファイルサーバの設定(NFS や HTTP)を作成

ノード登録を実施するにあたり，システム内のノード情報の管理方法として MySQL など DBMS を採用している OSS が多い．ノード登録コマンドで登録したノード情報は，インストール対象ノードごとに DBMS で管理し，上述したインストールサーバの設定時だけでなく，インストール中のノードのインストール状態管理に利用される．

図 1 に示すように，システム管理者はインストールサーバ上で，インストール対象ノードに割り当てたいホスト名や IP アドレスなどノードの情報を引数として，HPC インストールツールのノード登録コマンドを実行する．複数ノードを指定することができるノード登録コマンドもあるが 1 台のノード情報を指定することが一般的である．これによって，HPC インストールツールはノード登録コマンドで渡されたノード情報を自身が持つ DBMS に登録し，その情報をもとに DHCP サービスや TFTP サービスなどの設定を実施する．インストール対象ノードをネットワークインストールするために必要な設定を自動で実施することによって，インストール対象ノードを電源起動させるだけでネットワークインストールが開始される状態が整えられる．

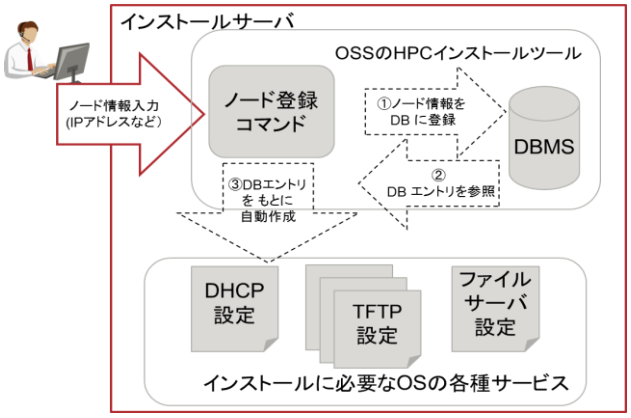

図 1　ノード登録処理の概要

3.2 リモート OS インストール

どの OSS の HPC インストールツールもリモートから OS をインストールする機能を有する．大規模システムを考慮した負荷分散の仕組みとして，インストールサーバを分離(階層構造化)させる仕組みを有する OSS も存在する．

リモートから OS をインストールする方法として以下の 2 つの方式があり，OSS ごとに採用する方式は異なる．

(1) パッケージインストール方式
インストール対象ノードは，インストールサーバからインストールするパッケージをダウンロードし，パッケージごとに適用処理(yum によるインストールなど)を実施する方式

(2) イメージ展開方式
インストール対象ノードは，インストールサーバ上に事前に作成したインストール対象ノード用の rootfs を圧縮したインストールイメージをダウンロードし，インストールイメージを展開する方式

図 2 に示すように，インストール対象ノードはネットワークブートすると自身の IP アドレスを取得するために，DHCP DISCOVER 要求を発行する．DHCP サーバとなるインストールサーバがこの要求を受け付ける．インストールサーバは，インストール対象ノードにインストール時に使用する自ノードの IP アドレスや TFTP サーバの IP アドレス，ネットワークブート用のブートローダ名を返す．

その後，TFTP サーバより，ブートローダを取得し，そのブートローダを使用してブートする．通常，CentOS や Fedora などで採用されている OS のインストーラである Anaconda の GUI にホスト名や IP アドレスなどを入力してインストールを実施するが，Anaconda には Kickstart ファイルと呼ばれるインストールする内容を定義して自動インストールする仕組みがある．これは，ネットワークブートで自動インストールする際にはその仕組みを使われることが多く，起動に必要なカーネルや起動イメージ(initrd)とあわせて，Kickstart のファイルパスを TFTP サーバから入手し，initrd を用いてブートする．

initrd によるブート後は，自動インストールするための

Kickstart ファイルやインストール部材をファイルサーバ(HTTP サーバなど)よりダウンロードしインストールを実施する.

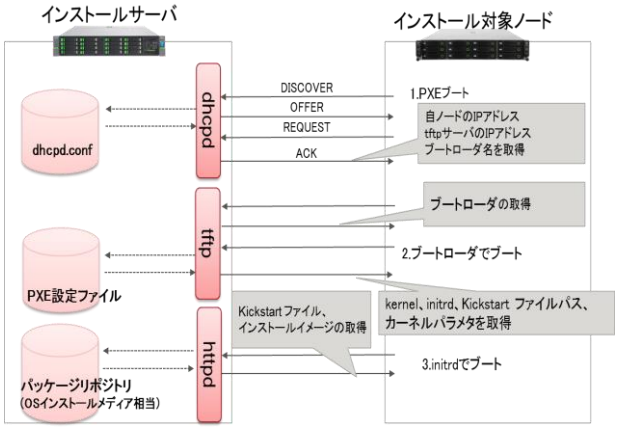

図 2　リモート OS インストール処理の概要

4. OSS の HPC インストールツールの比較

本論文では，HPC 分野で代表的な OSS の HPC インストールツールとして，Warewulf [2], Cobbler [3], xCAT [4] を掲げる．表 1 に各 OSS の HPC インストールツールの比較を示す.

各 OSS の HPC インストールツールは，表 1 のとおり，Warewulf および Cobbler は一部機能が不足しているが，xCAT は機能が充足している．ただし，xCAT においても，ノード登録の処理性能に課題がある.

1 台分のノードの設定ファイルが 0.1 秒で自動作成できると仮定すると，10,000 ノード分のインストールに必要な設定ファイルの作成に，1,000 秒(約 17 分)掛かることが目標時間になると考える．しかし，現状の OSS の HPC インストールツールは目標値から倍以上乖離している.

表 1　OSS の HPC インストールツールの比較

	Warewulf	Cobbler	xCAT
インストールサーバの設定自動化	機能あり	機能あり	機能あり
ノード登録単位	単一ノード	単一ノード	単一ノード/複数ノード
ノード登録方法	CUIコマンド	CUIコマンド	CUIコマンド
単一ノード登録処理時間	0m0.336s	0m0.277s	0m2.402s
10,000ノード登録処理時間	476m58.936s	49m42.408s	34m7.077s
リモートOSインストール	機能あり	機能あり	機能あり
パッケージインストール方式	機能なし	機能あり	機能あり
イメージ展開方式	機能あり	機能なし	機能あり
単一ノード登録処理時間	未測定	未測定	未測定

5. OSS の HPC インストールツールの問題点と課題

表 1 の OSS の HPC インストールツールを使用して超大規模システムとして 10,000 ノードをインストールすることを想定した場合に以下についてそれぞれ問題がある.

(1)　ノード登録処理

(2)　リモート OS インストール処理

本章ではそれぞれの問題について述べる.

5.1 ノード登録処理の問題

Warewulf と Cobbler のノード登録コマンドはインストール対象ノード数分，当該コマンドを逐次実行する必要がある．10,000 ノードを登録する場合に，コマンド実行あたり 1 秒掛かるとして 10,000 秒(= 2.8 時間)も必要となる．xCAT のノード登録コマンドは複数ノードを一括で登録することができるが，設定ファイルの作成処理に課題があり，処理に時間が掛かる.

また，ノード情報を DBMS で管理していることにより登録ノード数の増加に伴い，DBMS の検索や登録などのアクセス性能がオーバヘッドになり，ノード登録コマンドの 1 回の実行時間が長くなる．各 OSS の HPC インストールツールが採用する DBMS を表 2 に示す．設定に必要なファイル作成方法においても，ノードごとの設定ファイル作成が逐次的な処理となっている.

小規模なシステム構築の際には問題にならないノード登録性能が，大規模なシステムでは問題となり，リモート OS インストールをするための環境整備に長時間費やしてしまうことになる.

表 2　OSS の HPC インストールツールが利用する DBMS

	Warewulf	Cobbler	xCAT
利用 DBMS	MySQL, MariaDB	MySQL, CouchDB, MongoDB (利用しないことも選択可)	SQLite, PostgreSQL, MySQL, DB2

以降に，各 OSS の HPC インストールツールのノード登録処理における問題点を示す.

5.1.1 Warewulf の問題点

本節では Warewulf のノード登録処理の問題点について詳細を記載する．Warewulf は以下の 2 つのコマンドを実行することによってインストールサーバに必要な設定を実施する.

(1)　wwsh node new コマンドにより，1 台のノードずつ登録
　A) DBMS へノード情報を追加
　B) dhcpd.conf の再作成
　C) hosts の再作成

(2)　wwsh provision set コマンドにより，複数ノードを一括で OS 種別と紐づけ
　D) PXE 設定ファイルを作成

Warewulf では，DHCP サーバの設定である dhcpd.conf とホスト名の設定である hosts をノードごとに作成する(上記(1))．そして，上記(1)で作成したノード情報をもとに PXE 設定を実施している(上記(2))．上記(1)のコマンドにおいて，ノード登録数が多いほど，1 回のコマンド時間が増大するという問題がある．これは，ノード登録ごとに DBMS に登録されているノード情報の全エントリを書き出し，

dhcpd.conf や hosts を毎回再作成していることが原因である．

表 1 に示すように，全くノードが登録されていない状態で初回のノード登録をする場合は 0.3 秒で処理が完了するが，9,999 ノード登録した状態で最後のノードを登録する場合の処理時間は 5.7 秒と 19 倍に増大した(測定結果は，表 3 参照)．また，表 3 の結果より DBMS へ登録する処理に比べ，dhcpd.conf(上記(B))，hosts(上記(C)) の作成に時間が掛かっていることがわかる．これより，設定ファイル作成処理に問題があることがわかる．

表 3　Warewulf の 9,999 ノードを登録した状態で最後のノードの登録処理

処理	処理時間
通常(A,B,C実施)	0m5.762s
Aのみ実施	0m0.432s
A,B実施	0m2.769s
A,C実施	0m3.293s

5.1.2 Cobbler の問題点

本節では Cobbler のノード登録処理の問題点について詳細を記載する．Cobbler は以下の 2 つのコマンドを実行することによってインストールサーバに必要な設定を実施する．

(1) cobbler system add コマンドにより，1 台のノードずつ登録
   A) PXE 設定ファイルの作成
   B) ノード情報の生成
      DBMS を使用する場合は，DBMS へノード情報を追加するが，DBMS を使用しない場合は json 形式でノード情報用ファイルを作成する．
(2) cobbler sync コマンドにより，DHCP の設定
   C) dhcpd.conf の作成

Cobbler は DBMS を使用しないことが基本の設定となっているため，本論文では DBMS を使用しない設定で評価する．Cobbler では，上記(1)に示す PXE 設定ファイルをノードごとに作成し，上記(1)で作成したノード情報をもとに上記(2)で DHCP 設定を実施している．ノード登録ごとに処理量は変わらないため，登録ノード数に伴う処理時間の増加は生じない．そのため，9,999 ノードを登録した状態で最後のノードを登録した場合でも，初回ノード登録の処理時間と同じである(Warewulf の問題は発生しない)．

Cobbler のノード登録処理の問題として，1 台のノードを登録するごとに上記(1)のコマンドを逐次実行する必要があるため，全てのノードを登録するのに時間が掛かる問題がある．表 1 に示すように，1 台のノードの登録に 約 0.3 秒掛かるため，10,000 ノードを全て登録するのに約 50 分必要であることになる．また，他の OSS の HPC インストールツールと異なり，hosts は自動作成する機能を有さないため，システム管理者が別途作成する必要がある．

5.1.3 xCAT の問題点

本節では xCAT のノード登録処理の問題点について詳細を記載する．xCAT は以下の 4 つのコマンドを実行することによってインストールサーバに必要な設定を実施する．

(1) mkdef コマンドにより，複数ノードを DBMS へ一括登録
(2) makehosts コマンドにより，複数ノードを定義した hosts を一括作成
(3) makedhcp コマンドにより，dhcpd.conf を作成
   dhcpd.conf にネットワーク定義などの共通設定のみ作成
(4) nodeset コマンドにより，複数ノード分の PXE 設定と DHCP 設定を一括作成
   DHCP 設定は dhcpd.conf にノードのエントリを追加するのではなく，dhcpd.leases ファイルをノード数分の作成することで実現

xCAT のノード登録コマンドは複数ノードを一括で登録することができるため，Warewulf や Cobbler で発生する問題は生じない．しかし，一括ノード登録処理において，ノード数分の設定ファイルを逐次に作成しているため，処理に時間が掛かる問題がある．表 1 に示すように，10,000 ノードを一括で登録する場合に，約 34 分掛かる．10,000 ノードを登録する際の上記手続きごとの処理時間を表 4 に示す．表 4 より，(4)の複数ノード分の PXE 設定と DHCP 設定に最も時間が掛かっていることがわかる．これは，DBMS から全ノード情報を参照し，中間ファイルを作成した上で，ノード数分の設定ファイルを逐次に作成していることが原因である．

表 4　xCAT の 10,000 ノード登録時の処理時間

処理	処理時間
(1)	11m46.715s
(2)	2m10.680s
(3)	0m1.248s
(4)	20m18.434s

5.2 リモート OS インストール処理の問題

3.2 節で記述したリモート OS インストールの仕組みにあるように，インストール対象ノードのリモート OS インストールは，インストールイメージなどのインストール部材をインストールサーバよりダウンロードすることで実現される．大量のノードが一斉にリモート OS インストールが開始されると，インストールサーバに対してインストール部材のダウンロードが集中する．そのため，インストールサーバやネットワークの負荷により，インストール性能が低下する場合がある．

また，インストール方式によってノードのインストール時間が異なる．Cobbler で採用しているパッケージインストール方式では，インストールサーバからパッケージをダウンロードするための通信(HTTP アクセスなど)がパッケージ数分発生する．ダウンロード後はインストール対象ノード上でパッケージごとに適用処理(yum コマンド等)が実施

される．そのため，パッケージインストール方式では，インストールイメージをダウンロードして展開するのみのイメージ展開方式に比べ，インストールに時間が掛かる．また，パッケージインストール方式では，パッケージごとに適用処理を実施するため，パッケージの依存関係の問題(依存関係パッケージがダウンロードされていないなど)によりインストールが失敗する場合がある．そのため，システム管理者の作業の手戻り(依存関係パッケージの特定と再インストール)が発生し，システム構築時間に影響を与える場合がある．

これらのリモート OS インストール時間に関する問題点については今後検討する．

5.3 要件に対する課題

5.1 節および 5.2 節で述べた代表的な OSS の HPC インストールツールの問題点を踏まえて，第 2 章で述べた HPC インストールツールの２つの要件に対する課題を整理すると以下のとおりである．

要件 1. インストールサーバからリモート OS インストールをするための環境整備が手早くできること

課題 1. ノード登録に時間を要する問題

要件 2. HPC システム内のノード群を短時間でリモート OS インストールできること

課題 2. インストールサーバに対する負荷集中

課題 3. 処理に時間が掛かるインストール方式

このうち本論文では，試作評価まで完了している課題 1 の対処手法についてのみ詳細を述べる．

6. 課題を解決する HPC インストールツールの提案

本章では，前章で挙げた HPC インストールツールの課題 1 に対する対処手法として，ノード登録処理の並列処理化について述べる．

6.1 ノード登録処理の並列処理化

以下の 3 つの施策を実施することによって，ノード登録コマンドにおけるインストールサーバに必要な設定処理の高速化を図る．

(1) 複数ノードの一括ノード登録
(2) ファイル作成の効率化
(3) DBMS を利用しないノード情報の管理

以下，上記の施策について述べる．

6.1.1 複数ノードの一括ノード登録

ノード登録コマンドにおいて，コマンド引数に 1 台のノード情報のみを入力するのではなく，複数ノードのノード情報を記載したノード情報リストをファイルとして作成し，コマンド引数にノード情報リストのファイルパスを入力することによって，ノード数分の実行が必要であったノード登録コマンドの実行回数を 1 回にすることが可能である．

6.1.2 ファイル作成の効率化

インストールサーバで作成が必要な設定ファイルは，PXE 設定ファイルや Kickstart ファイルのようにノード数分作成が必要なファイルと，dhcpd.conf や hosts のように 1 つのファイルに全ノード情報を定義するファイルの 2 種類に分けられる．

ノード数分の設定ファイルを作成することが必要な PXE 設定ファイルや Kickstart ファイルは，1 つずつ作成するのではなく，ノード登録コマンドの引数で渡された複数ノードのノード情報をもとに，マルチプロセスで複数ノード数分のファイルを並列で作成する．この際，並列数はインストールサーバの CPU コア数をデフォルト値とする．

また，1 つのファイルに全てのノード情報を定義する dhcpd.conf や hosts については，ノード登録コマンド実行ごとに再作成せず，登録ノードのエントリのみを追加することで作成する．

6.1.3 DBMS を利用しないノード情報の管理

インストールサーバ上の設定ファイル作成処理において DBMS は必ずしも必要ではない．そのため，DBMS を使用せずノード登録処理を実現する．これにより，DBMS アクセス性能に起因したコストをなくし処理を高速化することができる．DBMS 活用の目的の一つである，インストール対象ノードのインストール状況の管理においては，インストールサーバ上に，インストール対象ノードのインストール状態を管理するインストール状態管理デーモンを設けることで実現する．

7. 提案 HPC インストールツールのアーキテクチャ

本章では，提案する HPC インストールツールである大規模システムインストーラのノード登録高速化設計を示し，その概要について述べる．

7.1 ノード登録高速化設計

大規模システムインストーラは，図 3 に示したように，システム管理者はインストールサーバにのみログインし，インストールに関するオペレーションをすることでシステム内のノード全てを構築することができる．システム管理者は，インストールサーバ上で一括ノード登録コマンドを実行する．システム管理者はシステム内の全ノードのホスト名や IP アドレスなどを記載したノード情報リストを作成し，一括ノード登録コマンドの引数としてノード情報リストを指定して実行する．これにより，一括ノード登録コマンドは引数で渡されたノード情報リストをもとに，インストール対象ノードのリモート OS インストールに必要なインストールサーバ上の設定を，インストール対象ノード数分一括して実施する．

一括ノード登録コマンドは，dhcpd.conf に全ノードのエントリ追加と，全ノード分の PXE 設定ファイルおよび，

Kickstart ファイルを自動作成し，リモート OS インストールのための環境整備を完了させる．これによってインストール環境が整備された後，システム管理者はインストール対象ノードを起動させることによって，自動で OS インストールが開始される．

ノード登録の際にインストールに必要な設定は次のように実施する．一括ノード登録コマンドでは，DHCP サーバ，TFTP サーバおよび，ファイルサーバ(HTTP サーバ)の設定ファイルを自動で作成するが，設定ファイルによって作成方法が異なる．1 つのファイルに全ノード情報を定義する dhcpd.conf や hosts については，インストール対象ノードを追加するごとに再作成せず，追加するインストール対象ノードのエントリのみを当該ファイルに追加する．ノードごとに異なるファイルを設定する必要がある PXE 設定ファイルと Kickstart ファイルについては，マルチプロセスで並列に作成する．この際，インストールサーバの搭載 CPU コア数分，ファイル作成処理のプロセスを生成することで並列にファイルを作成する．並列度については大規模システムインストーラが有する設定ファイルにおいてチューニングできる設計とする．

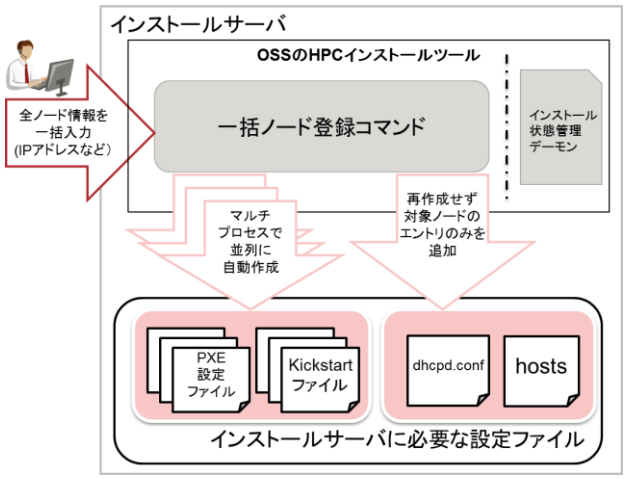

図 3　ノード登録高速化設計

7.2 提案するアーキテクチャの OSS への適用

7.1 節で述べたノード登録高速化設計(ノード登録コマンドの一括登録化，設定ファイル作成の効率化)は，容易に OSS の HPC インストールツールへ適用できると考える．

8. 試作評価

本章では，前章の 7.1 節で述べたノード登録高速化設計の試作評価について述べる．

8.1 評価方法

本評価では，7.1 節で述べたノード登録高速化設計を実装した一括ノード登録コマンドを用いて，10,000 ノードを登録する時間を測定し，Warewulf，Cobbler および，xCAT と比較検証した．一括ノード登録コマンドの概要は以下のとおりである．

・コマンドの引数にノード情報リストを渡し，複数ノードの同時登録
・dhcpd.conf, host ファイルはノード登録ごとに再作成せず，登録ノードのエントリのみを追加
・PXE 設定ファイル，Kickstart ファイルはマルチプロセスによりコア数多重で作成
・ノード登録処理において DBMS を使用しない

また，ノード登録において作成するファイルは以下とした．

・PXE 設定ファイル, Kickstart ファイル
　10,000 ノード分のファイルを作成
・dhcpd.conf, hosts ファイル
　1 つのファイルに全ノードのエントリを追加

比較検証する OSS の HPC インストールツールのノード登録コマンドと実行回数を表 5 に示す．

表 5　評価対象のコマンド実行方法

インストールツール	ノード登録コマンド	コマンド実行回数
Warewulf	wwsh node new コマンド	10,000回
	wwsh provision set コマンド	1回
Cobbler	cobbler system add コマンド	10,000回
	cobbler sync コマンド	1回
xCAT	mkdef コマンド	1回
	makehosts コマンド	1回
	makedhcp コマンド	1回
	nodeset コマンド	1回
大規模システムインストーラ	一括ノード登録コマンド	1回

8.2 評価環境

表 6 に示す環境で試作の評価を実施した．

表 6　評価環境

CPU	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz 16コア(4物理CPU * 4コア) HTなし
メモリ	24GB

8.3 測定結果

10,000 ノードをノード登録する性能を測定した結果と，大規模システムインストーラの一括ノード登録コマンドの多重度を変更した際の処理性能をそれぞれ表 7，表 8 に示す．

表 7，表 8 の結果より，OSS の HPC インストールツールのノード登録時間が約 34 分から 477 分掛かっていたのに対して，今回試作した大規模システムインストーラの一括登録コマンドでは約 2 分でノード登録が完了した．これらの結果について以下に分析する．

Warewulf は約 477 分掛かった．これは，5.1.1 節で示したように，1 台のノードを登録するたびに，登録したノード数分の dhcpd.conf，hosts を再作成する処理によるものである．

Cobbler は約 50 分掛かった．これは，5.1.2 節で示したように，ノードを登録するたびに設定ファイルを作成する処

理に時間が掛かったためである．

xCAT は約 34 分掛かった．これは，5.1.3 節で示したように，DBMS から全ノード情報を参照し，ノード数分の設定ファイルを逐次に作成している処理に時間が掛かったためである．

今回試作した大規模システムインストーラの一括ノード登録コマンドは，約 2 分と大幅に短縮した．これは，dhcpd.conf，hosts を再作成しない方式と，複数ノードを一括で登録する方式および，DBMS を利用せずノード数分の設定ファイル作成処理を並列化することによって，5.3 節で示した課題 1 を解決できることを示している．

また，設定ファイル作成の並列化の効果を確認するために，一括ノード登録コマンドをシングルプロセスで実行した場合の処理時間を測定した．1 多重で約 8 分掛かっていた処理が 16 多重にすることによって，約 2 分に短縮した．多重度分の時間短縮がされていない理由は，ファイル書き込みの I/O 処理がボトルネックとなっていたと考える．

表 7　ノード登録処理時間

インストールツール	ノード登録処理時間
Warewulf	476m58.936s
Cobbler	49m42.408s
xCAT	34m7.077s
大規模システムインストーラ	1m58.302s

表 8　登録処理の多重度評価

多重度	ノード登録処理時間
1	8m20.891s
16	1m58.302s

8.3.1 考察

6.1 節で示したノード登録処理の並列処理化の手法は高速化に有効であることを確認した．

全ノード情報を 1 つのファイルに定義する設定(dhcpd.conf，hosts)においては，ノード登録をするたびに，ファイルを再作成することなく当該ノードのエントリのみを追加する手法が高速化に有効であった．

ノードごとにファイルを作成する設定(PXE 設定ファイル，Kickstart ファイル)においては，ファイルを 1 つずつ作成するのではなく，マルチプロセスによりファイルを並列で作成することで，全てのファイルを作成する時間を短縮できた．

今後は，5.2 節で掲げたリモート OS インストール処理の問題解決方法についても検討を進める．今回提案した一括ノード登録コマンドは，大規模システムに対してのみ有用な手法ではなく，小中規模システムの構築の時間短縮にもなるため，今後評価をすすめ，HPC 分野で代表的な OSS の HPC インストールツールに本提案手法を適用していくことを検討していく．

9. まとめ

本論文では，超大規模 HPC システムのインストーラについて HPC 分野で代表的な OSS の HPC インストールツールがもつ課題のうち，ノード登録性能について述べ，それらを解決するアーキテクチャを提案した．

提案したアーキテクチャであるノード登録高速化設計の有効性を確認するために，試作評価を実施した．

試作では，複数ノードの一括登録，設定ファイル作成処理の並列化，DBMS を活用しないノード登録処理によって，大規模 HPC システムのリモート OS インストール環境整備の時間短縮を実現した．この手法は，HPC 分野で代表的な OSS の HPC インストールツールに容易に適用可能であると考える．今回の試作では，DBMS を活用しなかったが，実際の実装では活用することも考えうる．その場合には，アクセスが並列化される必要があり，その要件を満たす DBMS の選択や利用方法について，考慮を入れて設計する必要がある．

今回試作を実施しなかったリモート OS インストール処理の問題については，今後，検討していく．

謝辞　本論文の一部は，文部科学省「特定先端大型研究施設運営費等補助金（次世代超高速電子計算機システムの開発・整備等 ）」で実施された内容に基づくものである．

参考文献

[1] OpenHPC, https://openhpc.community/
[2] Warewulf, http://warewulf.lbl.gov/trac
[3] Cobbler, http://cobbler.github.io/
[4] xCAT, https://xcat.org/
[5] ポスト「京」プロジェクト, http://www.aics.riken.jp/fs2020p