6D-2

Condor VM ユニバースを利用した HPC Cloud の試作
A Prototype of HPC Cloud Based on Condor VM Universe

二木 邦尚 † 田中 良夫 ‡ 池上 努 ‡ 中田 秀基 ‡ 竹房 あつ子 ‡
Kunitaka Futatsugi Yoshio Tanaka Tsutomu Ikegami Hidemoto Nakada Atsuko Takefusa

1. はじめに

スーパーコンピュータの性能向上は目覚ましく，近い将来数百 TFlops から数十 PFlops 規模のスパコンが HPC インフラとして大学，研究所やデータセンターなど複数組織に次々と導入されていく事が予想される．多組織のスパコンを有効活用して科学技術や産業の発展を促進するためには，ネットワークを介してこれらのスパコンを容易に利用する環境の構築が必要である．グリッドコンピューティング [1] はそのための要素技術を提供するが，アプリケーション配備の際に組織毎に異なる計算環境への対応に手間がかかるといった問題がある．

本稿では，計算環境の非均質性を仮想化技術で吸収するクラウドコンピューティング的アプローチにより問題を解決し HPC クラウドを実現するシステムを提案する．本システムでは，計算に利用する仮想マシン (Virtual Machine 以下，VM) を必要数分起動させ，VM 上で並列計算を実行する．システムのプロトタイプをバッチジョブスケジューラ Condor を用いて実装し評価した結果，非均質性を隠蔽できることが示されたが，通信の仮想化による性能低下等，改善を要する部分も認められた．

2. VM 計算実行システム

提案システムは，VM 起動や HPC アプリケーション実行内容を定めたワークフローを解釈実行することで，空き資源の探索，計算に必要な VM の起動，VM 上での並列計算，計算完了後の VM 停止を行う．計算毎のこれら処理により HPC クラウドが実現される．以下では，システムの想定環境，計算プロセス，システム機能を示す．

2.1 想定環境

提案システムは広域ネットワーク上に複数の組織が個々に均質な計算機群 (以下，クラスタ) を保有しており，それらに VM 実行環境が構築されていることを想定する．各組織には外部との窓口役となるゲートウェイノード (以下，GW) が存在し，GW はクラスタ内の計算実行や VM 起動を管理する．クラスタ内ノード及び VM 間は相互に通信可能であるとし，かつ，ノード間で共有するファイルシステムの存在を仮定する．

2.2 計算プロセス

提案システムでは，VM イメージ及び VM 起動やアプリケーション実行内容を定めたワークフローをユーザが準備し，ワークフローを解釈実行する計算プロセスにより VM 上での並列計算を実現する．図 1 は計算プロセスを示したものである．以下に個々のプロセス内容を示す (項番号は図中番号に対応)．

(1)**VM イメージ作成**

OS やアプリケーション，アプリケーション実行に必要な各種ライブラリ・バイナリをインストールした VM イメージを作成する．イメージには VM 間で相互通信出来るような名前解決設定が施されるものとする．

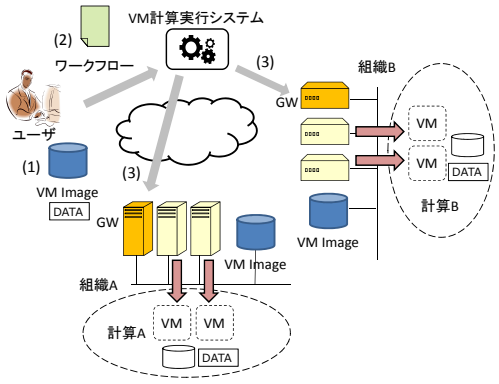

図 1: 計算プロセス

(2) **ワークフロー作成**

アプリケーション実行に必要な VM の指定 (VM 要求) と VM 上で実行する計算内容の指定 (計算要求) を含むワークフローを作成する．**VM 要求**には仮想マシンモニタの種類，ディスクイメージ，リソース要件 (VM 数，CPU コア・メモリ数) が指定される．ディスクイメージ情報にはイメージ場所と差分ディスク使用の有無が含まれ，イメージ場所をローカルまたはクラスタにすることでイメージ転送の有無が決定される．**計算要求**にはリソース要件 (CPU コア・メモリ数)，入出力，計算実行方法が指定される．計算は同一ワークフロー内の VM のみを計算資源として利用可能であり，かつ，消費するリソースは VM が提供するリソースを超えないものとする．

(3) **ワークフロー実行**

提案システムはユーザから入力されるワークフローを解釈し適切な資源を選択後 VM 起動依頼，計算実行依頼，VM 停止依頼を GW に送信する．

2.3 システム機能

システム内のワークフロー処理機能が VM 管理機能と計算実行機能を呼び出すことでワークフロー実行が実現される．

ワークフロー処理機能はユーザから入力されたワークフローを解釈し，VM 要求に基づき VM 管理機能で VM を起動し，起動確認後，計算要求に基づき計算実行機能により計算を実行する．そして計算完了確認後に VM 管理機能で VM を停止する．

VM 管理機能は VM 要求を入力として受け取り，各組織の GW へ資源情報 (仮想マシンモニタの種類，コア・メモリ) を問い合わせ，要求に適合するクラスタの GW へ VM 起動を依頼する．また VM 要求のイメージ情報からイメージ転送の有無を判断し必要に応じてクラスタ側へイメージを転送する．さらに VM 停止命令を解釈し適切な GW へ VM 停止を依頼する．

計算実行機能は計算要求，実行クラスタ (関連 VM 含む) 情報を入力として受け取り，実行クラスタの GW へ計算実行を依頼する．また，計算要求の入出力情報に基

†株式会社アルゴグラフィックス
‡独立行政法人産業技術総合研究所

表 1: Condor による実装機能

ワークフロー処理	DAGMan
VM 管理	VM・Parallel ユニバースジョブ
計算実行	Parallel ユニバースジョブ

表 2: 実験環境

仮想環境	Xen 3.0.3
実計算ノード (1 ノード)	Xeon E7540 2.0GHz * 4(計 24core), 64GB Memory, 265GB Disk, 1G-Ether, RHEL5.5
VM (2or1VM)	12or24core, 16or32GB Memory, 14GB Disk, 1G-Ether, CentOS5.5(PVM)

づき入力ファイル転送や計算結果転送を行う．

3. 実装

本システムの一部機能をバッチジョブスケジューラ Condor を用いて実装した．以下では Condor 概要及び実装機能を説明する．

3.1 Condor

Condor は分散計算システム上でハイスループットコンピューティングを実現させるスケジューリングシステムである [2]．もとは非グリッド環境上での利用が想定されていたが，グリッドミドルウェアを介したジョブ投入や Amazon EC2 へのインタフェースを備えるなど，グリッドやクラウド環境への対応も進められている．さらに，VMware，Xen，KVM 仮想環境の VM 起動制御をジョブとして実現する VM ユニバースや依存関係に基づきジョブ実行を行う DAGMan 機能を備えている．

3.2 実装機能

本実装では，VM 起動・停止，計算実行を Condor ジョブにより実現し，ジョブ間の依存関係を DAGMan で管理することでワークフロー処理を実現する．実装機能と Condor 概念の対応関係を表 1 に示す．

VM 管理機能による VM 起動は，クラスタ計算ノードに導入された Condor が VM ユニバースジョブ (以下，VM ジョブ) を実行することで実現する．計算実行機能は VM に導入された Condor が Parallel ユニバースジョブ (以下，計算ジョブ) を実行することで実現する．ワークフロー処理機能は VM ジョブと計算ジョブの依存関係に基づいたジョブ実行を DAGMan 機能で行うことで実現する．DAGMan 実行毎に一意に定められる DAGMan ジョブ ID を用いてワークフロー内の VM と計算を関連付けることで，計算毎の VM 独立性が保証される．

4. 評価実験

実装システムで計算を実行させ，基本的な動作確認と，非 VM 計算環境との比較を作業負荷や VM 利用に伴うオーバヘッドの観点から行った (ベンチマーク評価については別節に示す)．実験環境は表 2 の通りである．

4.1 動作確認と非 VM 環境との比較

ベンチマーク環境を導入した VM イメージをクラスタ側に設置後 DAGMan ワークフローを作成し，システムにワークフロー実行を指示するコマンドを発行すること

表 3: ベンチマーク内容

Application	HPL 1.0a(24 並列 MPI)
Compiler,Library	gcc 4.1.2, gsl 1.13, OpenMPI 1.4
MPI 通信レイヤ	共有メモリ or TCP

表 4: ベンチマーク結果 (Gflops)

	実ノード	VM	実ノード比 (%)
共有	22.8	21.1(1VM)	92.5
TCP	10.8	12.6(1VM)	116.7
		4.6(2VM)	42.6

で，VM 起動，計算実行，結果取得，VM 停止が自動で行われることを確認した．

非 VM 環境と比較し追加で発生する主な作業負荷に VM イメージ作成作業が認められた．イメージ作成にはアプリケーションに関する知識に加え OS やネットワークの知識を必要とする．VM 利用に伴うオーバヘッドの一つにワークフロー開始から計算開始までの遅延時間が認められ、80〜100 秒程度の遅延が観測された．遅延時間の 7 割弱を Condor スケジューリング時間が占め，残りが VM ブート時間であった．計算が長時間にわたる場合，遅延時間の全体への影響は軽微であると思われる．本結果はイメージ転送を伴わないケースのものであり，転送を要する場合，イメージサイズや通信環境に依存した転送時間が遅延時間に加算される．

4.2 ベンチマーク評価

VM 利用に伴う仮想化オーバヘッドの計算性能への影響を確認するため，表 3 のベンチマーク計算を実施した．計算は，1 実計算ノード内における 1VM での MPI 計算 (共有メモリ及び TCP 通信) と，2VM での MPI 計算 (TCP 通信) であり，結果を実ノード結果と比較した．

ベンチマーク結果を表 4 に示す．1VM に閉じた計算では実ノードとほぼ同等の性能が得られたが，2VM 間通信を伴う計算では実ノードと比較し半分以下の性能にとどまる結果となった．VM 間の仮想ネットワーク上での TCP 通信がボトルネックとなり性能低下が生じたものと思われる．

5. まとめ

本稿では計算環境非均質性を仮想化技術により吸収することで，分散環境における計算資源を有効活用した HPC クラウドの実現を図るシステムを提案した．プロトタイプ実装による評価実験の結果，非均質性を隠蔽した環境上での計算を単純なコマンド実行で行えることが示された．

今後は，未実装部分の実装を進めるとともに，イメージ作成負荷対策や性能改善の検討，実験の大規模化によるスケーラビリティ性確認などを予定している．

参考文献

[1] 合田憲人, 関口智嗣. グリッド技術入門. コロナ社, 2008.

[2] R. RamanHarput, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *7th IEEE International Symposium on High Performance Distributed Computing*, 1998.