2008－HPC－116 (5)
2008／8／5

仮想クラスタを用いた Data-Intensive Application 実行環境の性能モデル構築と最適化

佐藤賢斗[†] 佐藤 仁[†] 松岡 聡[†,††]

仮想クラスタ上で実行されるデータインテンシブアプリケーションを対象にしたマイグレーションによるデータアクセスの高速化手法を提案する．アプリケーションがアクセスするファイルの順序とサイズ、実行環境のノード間のネットワークバンド幅が既知であると仮定したときに、ファイルアクセスのコストと仮想マシンのマイグレーションからなるデータアクセスのコストに応じて、どのタイミングで、どのノードにマイグレーションするのかを決定する．我々は、この問題を仮想マシンの存在するノードとアクセス対象のファイルの組を頂点とし、データへのアクセスを辺とする有向非循環グラフ (DAG) として表現し、データアクセスコストを辺の重みとしたときの最短経路問題に帰着することで解く．提案手法をシミュレーションにより評価した結果、リモートからファイルへアクセスする場合に比べ最大で 40%、ファイルの存在する場所へ仮想マイグレーションを行う場合に比べ最大で 54%のファイルアクセスのスループット向上を確認した．

Model-based Optimization for Data-Intensive Applications on a Virtual Cluster

KENTO SATO,[†] HITOSHI SATO [†] and SATOSHI MATSUOKA[†,††]

We propose a model-based optimization algorithm that determines virtual machine migration strategies, i.e., which virtual machines should be migrated to which nodes, while minimizing I/O access costs on the assumption that the network bandwidth between nodes and the order, sizes and locations of target files are given. Our algorithm models this problem as a directed acyclic graph, where the vertex represents a location of a virtual machine when target files are accessed, the edge represents a flow of data access that includes a virtual machine migration and a remote I/O access, and the edge weight represents a cost of data access; we solves this problem as a shortest path problem that minimizes overall data access costs of target file accesses. Our simulation-based studies suggest that the proposed algorithm can achieve higher performance than simple techniques , such as ones that never migrate virtual machines or always migrate virtual machines onto the nodes that holds target files.

1. はじめに

近年、データインテンシブアプリケーションの実行環境としてグリッドの利用が実用的になりつつある．広域に分散した複数のサイトの資源を集約して用いることで、単一サイトでは実現できない大規模な計算環境、ストレージを実現する．このような環境でのデータ共有ではデータへのアクセスの高速化が重要な課題である．従来の手法 [1] では、データの複製作成やキャッシュを行うことで、ネットワークスループットによるアプリケーションの実行性能の低下を避けている．しかし、データインテンシブアプリケーションでは、アクセスの対象となるファイルの総サイズが非常に大きくなるため、複製作成のための転送時間や消費ストレージ容量などのオーバーヘッドが非常に大きくなる．

このような難しさは、仮想クラスタ [2][3] を用いることで解決できる．これは、動作中のアプリケーションの状態を維持したまま、仮想マシンをデータの所在するノード上へマイグレーションすることで、データへの高速なアクセスが実現できると期待されるためである．しかしながら、現在のノードから仮想マシンをマイグレーションしてデータへアクセスするべきか、どのノードへ仮想マシンをマイグレーションをするべきか、などの判断はアクセス対象とするファイルの総サイズ、仮想マシンに割り当てられたメモリサイズ、実行環境のノード間のネットワークスループットなどの様々なパラメタにより変化するため明らかではない．

我々は、仮想クラスタ上で実行されるデータインテ

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Informatics

ンシブアプリケーションを対象にしたマイグレーションによるデータアクセスの高速化手法を提案する．アプリケーションがアクセスするファイルの順序とサイズ、実行環境のノード間のネットワークバンド幅が既知であると仮定したときに、ファイルアクセスのコストと仮想マシンのマイグレーションからなるデータアクセスのコストに応じて、どのタイミングで、どのノードにマイグレーションするのかを決定する．我々は、この問題を仮想マシンの存在するノードとアクセス対象のファイルの組を頂点とし、データへのアクセスを辺とする有向非循環グラフ (DAG) として表現し、データアクセスコストを辺の重みとしたときの最短経路問題に帰着することで解く．

提案手法をシミュレーションにより評価した結果、リモートからファイルへアクセスする場合に比べ最大で 40%、ファイルの存在する場所へ毎回マイグレーションを行う場合に比べ最大で 54%のファイル読み出しのスループット向上を確認した．

2. 関連研究

広域分散環境でのデータへのアクセスの高速化において重要となるアイデアは、"いかに必要なデータをアクセス要求元の近くに置くか" という点である．既存手法では、ネットワークを経由するリモートアクセスを避けるために、1) ファイルの複製やキャッシュを作成する手法[1)4)] 2) ファイルの存在するノードにジョブを投入する手法[5)] などの戦略が採られる．しかしながら、このような手法を単純に適用しただけでは、いくつか問題がある．

ファイルの複製やキャッシュの作成する手法 どのファイルをどのノードへ複製するべきかの判断は、複製を作成することによる性能向上と消費されるストレージ容量との間にトレードオフの関係があるため、難しい問題である．アクセス頻度、ファイルサイズ、ネットワーク性能に応じて、適切な複製配置を実現しなければならない．たとえ、複製を作成したとしても適切な複製を選択しないと、アクセス性能が向上しないという問題もある．同様の議論は、キャッシュの作成においてもあてはまる．特にデータインテンシブアプリケーションでは、アクセスの対象となるファイルの総サイズが非常に大きくなるため、複製作成のための転送時間や消費ストレージ容量などのオーバーヘッドが非常に大きくなる．

ファイルが存在するノードにジョブを投入する手法 ジョブがアクセスするファイルが複数あり、それらが広域に分散した異なるノードに存在する場合、リモートアクセスの発生は避けられず、オーバーヘッドとなり、適時ファイルの複製やキャッシュを行う必要が発生する．

これらの問題は仮想マシンのマイグレーションを用いることで、動作中のアプリケーションの状態を維持したまま、データの存在するノード上へマイグレーションすることで、データへの高速なアクセスが実現できると期待される．

近年、グリッド環境におけるジョブの実行環境として、仮想クラスタの利用が数多く提案されている．仮想クラスタ[2)3)] は、仮想化技術を用いたコンピューティングクラスタであり、仮想マシンを計算ノードとして利用することで、下位の物理マシン環境とは独立に柔軟な環境を構築することができる．我々の提案手法は、仮想クラスタ上でのデータインテンシブアプリケーションの実行を対象としている．このような環境で仮想マシンやプロセスをマイグレーションすることにより、性能向上を行っている研究がいくつかある[6)]、しかし、これらは、データインテンシブアプリケーションの実行をサポートしていない．

3. 提案手法

我々は、仮想クラスタ上で実行されるデータインテンシブアプリケーションを対象にしたマイグレーションによるデータアクセスの高速化手法を提案する．アプリケーションがアクセスするファイルの順序とサイズ、実行環境のノード間のネットワークバンド幅が既知であると仮定したときに、ファイルアクセスのコストと仮想マシンのマイグレーションからなるデータアクセスコストに応じて、仮想マシンをどのタイミングで、どのノードにマイグレーションするのかを決定する．

3.1 仮想マシンを利用したデータアクセス

アクセス対象のデータが存在するノードの近くへ仮想マシンをマイグレーションすることにより、リモートのデータアクセスと比較して、性能の向上が期待される．今回、我々は仮想マシン・モニタとして Xen[9)] を対象とする．Xen では Stop-and-Copy 法とライブ・マイグレーションと呼ばれる Pre-Copy 法の 2 種類をサポートしている[7)]．前者はマイグレーションが実行されると仮想マシンをサスペンドし、メモリ内容を移動先へ転送、完了後リスタートさせる方法をとる．このため仮想マシンのメモリサイズの転送でマイグレーションが完了する．後者は仮想マシンを動作させたままメモリの内容を転送し、その間に生じたダーティ・ページを繰り返し転送した後、仮想マシンをサスペンドさせ残りのページを転送し、完了後リスタートさせる方法をとる．このためダーティ・レートの高いアプリケーションを実行した場合、マイグレーション時間の増加を引き起こす．今回、我々は簡単のため Stop-and-Copy 法を対象として、ファイルアクセスのコストと仮想マシンのマイグレーションのコストからなるデータアクセスのコストモデルを構築する．

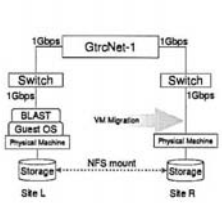

図 1　モデル構築環境

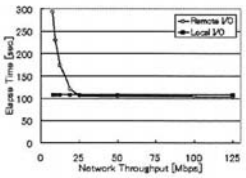

図 2　POSIX I/O による
シーケンシャル読み出し時間

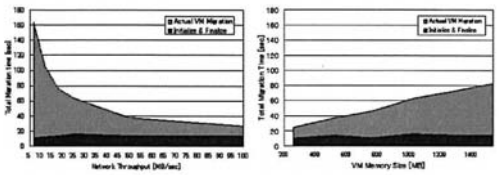

図 3　マイグレーション時間の内訳
VM Memory Size:1024MB

図 4　マイグレーション時間の内訳
Network Throughput:1024MB

3.2　データアクセスのコストモデル

データアクセスのコストモデルを構築するために、東京工業大学松岡研究室 PrestoIII クラスタ 2 ノードをネットワーク・エミュレータ GtrcNet-1[8] を介して接続 (1Gbps) した図 1 のような環境で予備実験を行った．一方のノードのローカル・ディスクに VM イメージを、他方のノードにアクセス・データを用意し、双方のノードがそれらを NFS マウントしている．クラスタのノードはメモリを 2GB、CPU は Opteron250(2.4GHz) × 2、NIC は Broadcom Corporation NetXtreme BCM5704 を搭載し、カーネルは 2.6.18-xen、Xen のバージョンは 3.1.0 を使用した．

まず、リモートのノードに存在するファイルへアクセスした際の read アクセス時間のコストモデルを構築するための実験を行った．実験は、2GB のファイルの全領域をシーケンシャルに read するという操作をネットワークのスループットを変えて行った．図 2 にネットワークのスループットとファイルの読み出しに費やされた時間の関係を示す．ネットワークのスループットが低い場合は、その部分がボトルネックとなり読み出しの実行時間に影響を与えているが、ネットワークのスループットの性能が高くなるにつれてディスク、ファイルシステム性能などに起因するローカルノード上でのファイルアクセスのスループットの低下がボトルネックとなることを確認した．この結果から、read アクセスの I/O サイズ (io_size[MB]) に対するリモートの read アクセスの実行時間 (r[sec]) をファイルアクセスのスループット (H[MB/s]) を次式で表す．

$$r = \frac{io\_size}{H}$$

$$H = \min\left\{h, d\right\}$$

ここで、h をネットワークのスループット (MB/s)、d をローカルノード上でのファイルアクセスのスループット (MB/s) とする．

次に、仮想マシンのマイグレーションに要する時間のコストモデルを構築のための実験を行った．図 1 の環境で、バイオインフォマティクスの分野で広く利用されている相同性検索ツールである BLAST[10] を実行させ、実行途中で仮想マシンのマイグレーションを行った．アクセスの対象とするファイルの総サイズは 2.03GB とし、ネットワークのスループットと仮想マシンのメモリサイズを変化させて実験を行った．図 3 は、仮想マシンのメモリサイズを 1024MB に固定し、ネットワークのスループットを変化させたときの仮想マシンのマイグレーションの経過時間を表し、図 4 は、ネットワークのスループットを 25MB/sec に固定し、仮想マシンのメモリサイズを変化させたときの仮想マシンのマイグレーションの経過時間を表す．図 3、図 4 で、Actual VM Migration の部分は、ネットワークのスループットと仮想マシンのメモリサイズに依存する項目であり、Initialize & Finalize の部分は VM サスペンド、リジュームなどの処理の部分で、ネットワークのスループットや仮想マシンのメモリサイズに依存せず定数であった．この結果から、仮想マシンのマイグレーションに要する時間のコストモデル (m[sec]) を以下のように構築した．

$$m = \frac{v}{h} + C$$

ここで、v は仮想マシンのメモリサイズ (MB)、h はネットワークのスループット (MB/s)、C(sec) は Initialize & Finalize の時間を表す定数とする．

以上のファイルアクセスのコストモデルより、仮想マシンのマイグレーションの最適化を行う．

3.3　仮想マシンマイグレーションの最適化

ノード間のネットワークスループットとファイルシステムのスループット、ジョブがアクセスするファイルサイズとそのファイルが存在するノード、ジョブが投入されたときの仮想マシンの位置とメモリサイズからファイルアクセスの時間が最小になるように、仮想マシンを移動させる．その手順の概要を図 5 に示す．3 つのフェーズからなり、Phase1 では、入力された値から全ノード間のマイグレーション時間とファイルアクセス時間を算出 (Model Applying) し、Phase2 では、その値から仮想マシンの位置とアクセスデータの組を頂点、ファイルへのアクセスを辺とする有向非循環グラフ (DAG) を生成する (DAG Making)．Phase3 では、ファイルアクセス時間を辺の重みとしたときの最短経路問題を解くことで、ファイルへのアクセス時間を最小にする仮想マシンの位置を決定する (Shortest Path Search)．次にそれぞれのフェーズについて詳し

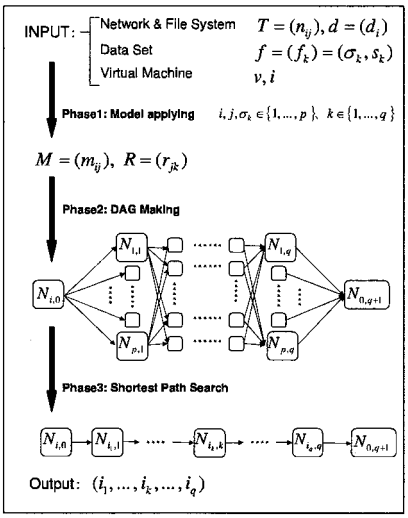

図 5 提案手法の手順

く説明する．以下では計算ノードのことを単に"ノード"、後に説明する有向グラフ上のノードのことを"頂点"と呼び区別する．

Phase1: Model Applying Phase2 で DAG を生成するための準備として 3.2 節で構築したコストモデルから、ファイルの読み込み時間 (r_{jk}) とマイグレーションの時間 (m_{ij}) を計算する．まず実行するジョブが k 番目にアクセスするファイルをノード j から読み出すに要する時間 r_{jk} を実行するジョブが k 番目にアクセスするファイルの読み込みサイズ $s_k(MB)$、そのファイルが存在するノード σ_k、ノード j のローカルファイルアクセスのスループット d_j(MB/s)、ノード $i-j$ 間のネットワークスループット h_{ij} を用いて、次式で表す．

$$r_{jk} = \frac{s_k}{\min\left\{h_{j\sigma_k}, d_{\sigma_k}\right\}}$$

次にノード i からノード j へのマイグレーション時間 m_{ij} はジョブを実行させる仮想マシンのメモリサイズ v(MB)、h_{ij}、次式で表わすことができる．

$$m_{ij} = \frac{v}{h_{ij}} + C\ (i,j \in \{1,\ldots,p\})$$

ただし $i=j$ のときは $m_{ij}=0$ とする．ここで、p はネットワーク上の全ノード数を表し、C は v や h_{ij} に依存しない定数である．q はそのアプリケーションがアクセスするファイルセットの個数を表す．

Phase2: DAG Making m_{ij}、r_{jk} から最短経路問題へ帰着するための DAG を生成する．まず頂点 N_{ik} はノード i 上で k 番目のファイルを読み出すことを意味し、ファイルアクセス時間を表す頂点 X-Y の辺の重み $w(X,Y)$ を次式で表す．

$$w\left(N_{i,k}, N_{j,k+1}\right) = m_{ij} + r_{j\sigma_{k+1}}$$

ただし $k=q$ のとき、

$$w\left(N_{i,q}, N_{0,q+1}\right) = 0$$

ここで得られた DAG 上を頂点 $N_{i,k} \rightarrow N_{j,k+1}$ へ辿ることは、「ノード i から k 番目のファイルを読み出した VM をノード j へ m_{ij} 秒かけてマイグレーションし、ノード j から k+1 番目のファイルを $r_{j,\sigma_{k+1}}$ 秒かけて読み込む」ことを意味する．$i=j$ のときはマイグレーションをせずリモートから k+1 番目のファイルを読み込むことを意味し、また $m_{ij}=0$ となるので $r_{j,k}$ 項だけが残る．頂点 $N(0,q+1)$ は最短経路問題へ帰着するためダミー頂点である．これを追加し、DAG の終点を一意に定めることで最短経路問題に帰着できる．

Phase3:Shortest Path Search 生成した DAG の $N(i,0)-N(0,q+1)$ 間の最短経路をダイクストラ法を用いて求める．このとき得られた最短経路を $N_{i,0}, N_{i_1,1}, \ldots, N_{i_k,k}, \ldots, N_{i_q,q}, N_{i_0,q+1}$ とすると、k 番目のファイルをノード i_k 上で読み込むことで全てのファイルを最小の時間で読み込むことができ、そのときの時間は得られた経路の長さになる．

4. 実　　験

シミュレーション環境としては図 6 に示すように仮想マシンはノード 1 上で待機しており、1000、500、200、10MB のデータは各サイトに分散されている環境を想定する．例えばノード 13 上には 1000MB、ノード 16 上には 10MB のデータがローカルディスク上に存在することを意味する．また、ネットワークのスループットはサイト内では一律 125MB/s であるがサイト間では 100MB/s もしくは 10MB/s となる．この環境で表 1 に示す値を用いてシミュレーションを行った．また、ローカルノード上でのファイルアクセスのスループットをここではディスクのスループットとし全てのノードが同じ性能であるとする $(d_i = d(\text{一定}))$．ファイルセット:$f=(f_k)=((s_k,\sigma_k))$ はファイルのサイズと位置をアクセスする順に並べたものである．また、マイグレーションのモデル式中の定数 (C) は図 3、4 の実験で得られたデータの平均値を元に $C=13$

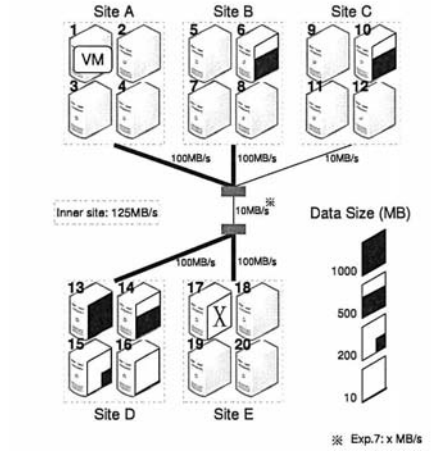

図 6 シミュレーション環境

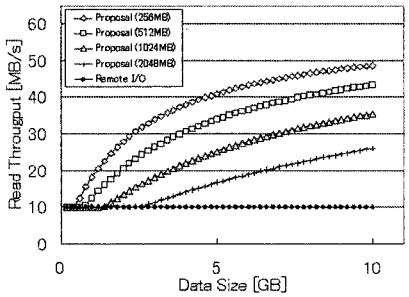

図 7 データサイズによる read スループットの変化

とした. また、マイグレーションを行ったときのファイルアクセスのスループットは次式で表される.

$$\frac{\sum_{k=1}^{p} s_k}{T_r + T_m}$$

ここで、T_r はファイルセットを全て読み込むのに要した時間の合計を表し、T_m は複数回行ったマイグレーションの時間の合計を表す. マイグレーションを行わない場合は $T_m = 0$ とする.

4.1 予備評価

Exp.0 ではノード 1 上の仮想マシンからノード 17 上のファイルを読み込ませるシミュレーションを行った. ディスクのスループットは d=60MB/s とした. まず、サイト ABC-サイト DE 間のネットワークスループットを 10MB/s に固定し、ノード 17 上のデータサイズと仮想マシンのメモリサイズを変えて、提案手法を適用した場合におけるファイル読み出しのスループットの変化を図 7 に示す. Remote I/O は常にノード 1 上からノード 17 上のファイルを読み込ませたときのスループットを表す. このことから提案手法を用いることにより、データサイズの変化に応じてマイグレーションを行い、リモートからのファイルアクセスより高いスループットが出ることがわかる.

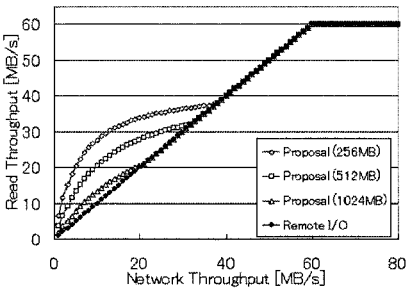

図 8 ネットワークスループットによる read スループットの変化

次に、ノード 17 上のファイルサイズを 2000MB に固定し、サイト ABC-サイト DE 間のネットワークスループットと仮想マシンのメモリサイズを変えて同様の実験を行った場合のファイルアクセスのスループットの変化を図 8 に示す. この結果から提案手法を用いることにより、ネットワークのスループットが低い環境でも、マイグレーションを行うことによって、リモートアクセスより高いスループットを維持できることがわかる.

4.2 仮想マシンの移動経路の変化

仮想マシンの移動経路の変化とその性能を表 2 に示す. Remote I/O はマイグレーションを行わずリモートからのアクセスのみを行った場合、Migration I/O は常にアクセスデータが存在するノードへマイグレーションをした場合、Proposal は提案手法を適用した場合のファイルアクセスのスループット (MB/s) を表す. Exp.2 では仮想マシンのメモリサイズが大きいためマイグレーションを行わずにリモートからファイルへアクセスした. Exp.1、Exp.3 ではファイルが存在するノードへマイグレーションを行っているが、Exp.4、Exp.5、Exp.6 ではマイグレーション先とアクセスするファイルが存在するノードに違いがある. これはディスクよりネットワークのスループットの方が高い場合、リモートからファイルへアクセスしてもローカルアクセスと同じスループットになるようにモデル式を立式したためである. 実際にはファイルアクセス時間が等しい複数の経路が得られた.

この結果から提案手法を用いることにより、仮想マシンのメモリサイズ、アクセスするファイルの位置、サイズ、アクセス順序やディスクのスループットに応じて、適切に仮想マシンの移動させるとこにより、リ

表 1 評価項目

Exp.#	仮想マシンのメモリサイズ (v[MB])	データセット:$f = (f_k) = ((s_k, \sigma_k))$	ディスク性能 (d[MB/s])
Exp.0	X	$((X, 17))$	60
Exp.1	256	$((1000, 13), (500, 14), (200, 15), (10, 16))$	20
Exp.2	1024	$((1000, 13), (500, 14), (200, 15), (10, 16))$	20
Exp.3	1024	$((1000, 13), (500, 14), (200, 15), (10, 16))$	60
Exp.4	256	$((500, 6), (500, 10), (500, 14))$	60
Exp.5	256	$((500, 6), (500, 10), (500, 14), (500, 6), (500, 10), (500, 14))$	60
Exp.6	256	$((500, 6), (500, 10), (1000, 13), (500, 6), (500, 10), (1000, 13))$	60

表 2 評価結果

Exp.#	Remote I/O (MB/sec)	Migration I/O (MB/sec)	Proposal (MB/sec)	$(i_1, \ldots, i_k, \ldots, i_q)$
Exp.1	10	10.1	**13.8**	**(13,13,13,13)**
Exp.2	10	6.5	**10**	**(1,1,1,1)**
Exp.3	10	8.2	**11.9**	**(13,13,13,13)**
Exp.4	13.9	12.7	**16.7**	**(1,9,13)**
Exp.5	13.9	11.6	**14.3**	**(1,1,1,1,9,13)**
Exp.6	12.6	14.5	**17.6**	**(1,9,13,13,13,13)**

read スループット [MB/s]

モートからファイルへアクセスする場合に比べ最大で 40%、ファイルの存在する場所へ毎回マイグレーションを行う場合に比べ最大で 54%のファイルアクセスのスループット向上を確認した.

5. おわりに

我々は仮想マシン上で実行されるデータインテンシブアプリケーションを対象にしたマイグレーションによるデータアクセスの高速化手法を提案しその評価を行った. 評価の結果、提案手法を用いることにより、リモートからファイルへアクセスする場合に比べ最大で 40%、ファイルの存在する場所へ毎回マイグレーションを行う場合に比べ最大で 54%のファイルアクセスのスループット向上を実現した.

今後の課題としては、計算ノードの CPU、メモリを考慮したマイグレーションや複数の仮想マシン間の配置を考慮した最適化が挙げられる.

謝辞 本研究の一部は科学研究費補助金特定領域研究 (18049028) の支援によって行われた.

参考文献

1) Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, and Satoshi Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pp. 102 – 110, 2002.
2) Hideo Nishimura, Naoya Maruyama, and Satoshi Matsuoka. Virtual clusters on the fly - fast scalable and flexible installation. In *Proceedings of the 7th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 549–556, 2007.
3) I.Foster, et al. Virtual clusters for grid communities. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, pp. 513– 520, 2006.
4) Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, Vol. 38, No. 1, p. 3, 2006.
5) Srinath Shankar and David J. DeWitt. Data driven workflow planning in cluster management systems. In *Proceedings of the 16th IEEE International Symposium on High Performance Distributed Computing*, pp. 127–136, New York, NY, USA, 2007.
6) Masaki Tatezono, Hidemoto Nakada, and Satoshi Matsuoka. Mpi environment with load balancing using virtual machine. In *Symposium on Advanced Computing Systems and Infrastructures SACSIS*, pp. 525–532, 2006.
7) Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny, Eric July, Christian Limpach, Ian Pratt, and Andrew Wareld. Live Migration of Virtual Machines. In *NSDI*, 2005.
8) Y. Kodama, T. Kudoh, R. Takano, H. Sato, O. Tatebe, and S. Sekiguchi. GNET-1:Gigabit Ethernet Network Testbed. `http://projects.gtrc.aist.go.jp/gnet/`.
9) Xen Community. http://xen.xensource.com/.
10) NCBI BLAST. `http://www.ncbi.nlm.nih.gov/BLAST`.