大学院中間発表配付資料 平成 19 年 4 月 13 日

データ転送を最適化する分散タスクスケジュラー
A Distributed Task Scheduler Optimizing Data Transfers

田浦研究室　　電子情報学専攻　修士 2 年 56428 高橋 慧

Abstract

A task scheduler, which distributes many serial tasks onto machines, enables users to fully utilize distributed environments. To schedule data intensive applications such as natural language processing or data mining, a scheduler needs to consider original data positions and to plan efficient file transfers. Some existing task schdulers plan data transfers by using static bandwidth values, but the bandwidth between two hosts significantly changes when mutiple data aretransferred at the same time.

We propose a schduling algorithm which optimizing data transfer time by using network topology and bandwidth information. For a given schedule, a efficient data transfers are planned to maximizing throughput of data transfers on each node by using Kruskal's algorithm and linear programming. Based on this throughput value, the task transfer scheule is improved to maximize it by using heuristics such as GA.

The implementation of this algorithm is ongoing, and the evaluation will be done by using real data-intensive applications. The data transfer time and total completion time will be compared to a scheuling algorithm which does not use the network topology.

1　はじめに

1.1　背景

コンピュータの性能向上、特に記憶容量の飛躍的な増加によって、これまで望めなかった大規模な処理が可能になっている。例としては、Web 上のドキュメントに対する言語処理・遺伝子解析などが挙げられる。これらは処理自体は単純であっても、大規模なデータに対して処理を行うことで、これまで得ることが出来なかった成果が上がっている。これらをデータ・インテンシブなアプリケーションと呼び、データサイズはテラバイトからペタバイトに達するので、並列処理が不可欠になっている。

このようなアプリケーションは多くの場合処理の局所性が高いので、逐次タスクの集合として表すことが出来る。一般には並列プログラムを記述するのは容易でないが、このように処理全体が逐次タスクの集合で表されている場合、分散タスクスケジュラー [1] と呼ばれるシステムを用いることで、簡単に並列処理を行うことができる。これは多数のノードが接続されている環境で、一群の逐次タスクをノードに割り振り、かつ必要なファイルを適宜転送するものである。基本的にはタスクを遠隔のノードで実行し、必要に応じてファイルを転送できるシステムがあればよい。

並列処理を行う環境として、グリッドと呼ばれる分散環境が着目されている。これは多数のノードをネットワークを用いて接続したもので、莫大な計算力を簡単に得られる反面、ノードは地理的に離れていることが多いので、ノード間のバンド幅は場所により 1Gbps から 10Mbps 程度と大きく異なる。このような環境でデータ・インテンシブなアプリケーションをスケジューリングする場合は、全体の実行時間に占めるデータ転送時間が大きいので、効率的なデータ転送計画を立てることが重要になる。

まず、データ転送を減らすようなスケジューリングが必要になる。タスク実行に必要なデータがあるノードにあるときは、可能ならそのノードで実行すると、データ転送が不要になる。データ転送が必要になる場合でも、元々のデータを持つノードからなるべく近いノードで実行すると、データ転送時間は短縮される。

次に、あるタスクスケジュールに対して転送を最適化することが考えられる。複数のノードが同一のファイルを必要としている場合には、それらのノードを一列に並べてパイプライン転送を行うことで効率的な転送を行うことが出来る。初めから複数のノードがそのファイルのコピーを持っている場合には、複数のソースをうまく用いることでバンド幅を向上できる可能性がある。また、あるリンクを複数の転送が共有しているとき、多くのノードが必要としているデータの転送は重要であると考えられるので、より多くの

バンド幅を与えることで全体のファイル転送のスループットが向上する。

1.2 関連研究

データインテンシブなアプリケーションのスケジューリングに関しては、既にいくつかの研究がある。まず、GrADS プロジェクト [4] では、データ・インテンシブな環境に対応したスケジュラーの開発を行っている。処理系として賢いスケジュラーを実装する上で必須になるのが、アプリケーションの実行時間やファイルサイズを予測することである。GrADS では、まず各ノードの性能を、整数演算性能・浮動小数点演算性能・CPU クロック・キャッシュサイズ・メモリクロック・メモリサイズなどを用いてパラメータ化し、また実際のプログラムを小さなデータで実行することでアクセスパターンを測定し、タスクの実行時間を推測する。またファイル転送の時間についても、各ノード間のバンド幅・レイテンシを測定することで予測し、より短い時間で実行できるノードにタスクを割り当てる。

しかし、この研究ではノード間のバンド幅は固定されているものとされており、データ転送時間が総実行時間に大きく影響するような環境下で実際に多数の転送が発生した場合、これらの転送がリンクを共有することにより、予め計算に用いたバンド幅では転送が行えないと考えられる。4 つの転送が一つのリンクを共有している場合、単純計算では各転送は 25%のバンド幅しか用いることができず、転送時間は大幅に長くなってしまう。

町田ら [5] は、同じデータを必要としているノードが複数あるとき、そのデータのレプリカを作成することでデータ転送量を削減する試みを行った。これにより、実験に用いられたタスクでは NFS を用いてデータ転送を行った場合に比べて約半分の時間で転送が完了した。この研究はデータ転送計画を賢く立てることで転送時間の短縮が図られることが具体的に示されている点で重要だが、リンクを共有している場合の議論はされていない。また実験環境は 1 クラスタで均質なバンド幅を持つため、ネットワーク的に不均質なグリッド環境では各リンクのバンド幅を考慮したスケジューリングが必要になると考えられる。

1.3 本研究の貢献

本研究では、データインテンシブなアプリケーションに対して、各ノードに割り当てられたタスクに必要なデータが最も早く到着するようなスケジュールを立てることにより、タスク群全体の処理時間を最小化するようなスケジューリングアルゴリズムを提案する。このアルゴリズムにおいて与えられるのは、以下の三つである。

1. タスク群及び必要としているデータ
2. データの位置
3. ネットワークトポロジ・バンド幅

ここで、以下の二つが求められる。

1. タスクのスケジュール
2. データ転送のスケジュール

1 は、ノードへのタスクの割り当てであり、2 はファイル転送をどのノードからどのノードに向けて行うかという計画である。

タスク群が与えられたとき、タスクの実行コストが既知であれば、全てのタスクを一度に各ノードに割り当ててしまう方法も考えられる。これを静的スケジューリングと呼ぶが、一般にタスクの実行時間を実行前に予測することは大変難しい。実行前にプロファイリングを行えばある程度の予測は可能になるが、本研究では全てのタスクについてプロファイリングを義務付けるようなアプローチは取らず、各ノードには 1 回のスケジューリングにつき 1 つずつのタスクを割り振る、動的なアプローチをとる。

タスク自体の実行時間の予測が難しいのに対し、ネットワークのトポロジ及び各リンクのバンド幅が既知であれば、あるデータの転送時間などのネットワークの挙動は比較的正確に予測できる。このため、データ転送時間を最適化するようなタスク割り当てを行い、実際のデータ転送もマルチキャストなどを用いて効率的に行えば、タスク群の処理時間も最適化することができる。

データ転送時間を最適化するために着目することは、同じデータを多数のノードが必要としている場合にマルチキャストを用いること、マルチキャストのソースとして最も適したものをクラスカルのアルゴリズムを用いて決定すること、各データの転送を、線形計画法を用いて求めたバンド幅で制限して行うことである。

動的なスケジューリングであるため、新たなタスクは随時追加することができる。また、バンド幅が変化した場合やあるデータの転送が終わった場合は、転送計画のみを再構成する。これにより、実行中の環境変化にも対応したスケジューリングを行うことが出来る。

2　タスクのスケジューリング

タスクのスケジューリングは発見的手法を取るが、評価する基準としてデータの転送時間を用いる。あるタスクスケジュールが与えられたとき、各ノードにつきタスクが割り当てられ必要なデータが定まるが、これらのデータが届くスループットを基準として用いる。ローカルディスクにある場合は、ディスク読み出しのスループットを用いるものとする。全てのノードについてスループットの和を計算し、これが最大になるようなスケジュールを求めることを考える。

まず初期タスクスケジュールを定め、これについてデータ到着のスループットが最大になるようなデータ転送計画を決定する。その際にネットワークトポロジ中の各リンクの利用率が求められるが、その際に 100%用いられているリンクは全体のスループットを制限している可能性が高いので、そのリンクに関わるデータを必要としているタスクの割り当てを優先的に変更し、新しいタスクスケジュールの候補を作る。これについて再びデータ転送を計画し、スケジュールを改善していく。一定時間このルーチンを実行したら、今までで最も大きいスループットを与えたスケジュールを採用して実行する。タスクスケジュールを改善していくアルゴリズムは、遺伝的アルゴリズムや焼き鈍し法などを比較の上で用いる予定である。

ノードでタスクの実行が完了すると、再スケジューリングが必要になる。これは、一定数のノードでタスク実行が終了した場合か、タスクが終了してから一定時間が経過した際に行われる。再スケジューリングの際には、現在行われているデータの転送も含めて計画を行い、スケジューリング完了後は各転送に割り当てられるバンド幅は変化する。

3　データ転送の最適化

3.1　全体のアルゴリズム

あるタスクのスケジュールが与えられると、どのノードにどのタスクが割り当てられ、そのタスクがどのデータを必要とされているかが定まる。また、初めからどのノードがどのデータを持っているかの情報は与えられており、ネットワークトポロジや各リンクのバンド幅も与えられているものとする。このとき、各ノードに到着するデータのスループットを最大にするようなデータ転送計画を立てる。

まずそれぞれの転送がリンクを共有せず、バンド幅を 100%使えると仮定してデータの転送を計画する。ソースが一つしか無い場合には転送ルートは一通りに定まるが、複数のノードがデータのコピーを持っている場合には、どのソースからファイルを転送するかを決定する必要がある。また、複数のファイルを一つのソースから転送する場合にはパイプラインを用いたマルチキャストを行う計画をクラスカルのアルゴリズムを用いて立てる。この計画は、複数のファイル転送が同時に存在することで最適ではなくなる可能性もあるが、それについては後程修正する。

ここで、どのノードがどのノードからファイルを転送するかが求まったので、次にそれぞれのファイル転送にどれだけのバンド幅を割り当てるかを決定する。これは、多くのノードが関係するデータ転送を優先することで、全体のスループットを向上させることができるという原理に基づいている。計算には線形計画法を用いる。

これにより、各ノードに到着するデータ量と、全てのリンクを通過するデータ量が求まった。ここで、先ほど立てたマルチキャストの計画が最適になっているかを再検討し、計画を変更することでバンド幅が改善される場合はそのように変更する。この操作を全てのデータ転送について行い、再度線形計画法を用いてバンド幅の割り当てを行う。

以上のアルゴリズムを整理すると、以下のようになる。

1. 各データ転送について、そのデータを必要としている各ノードに対し、クラスカルのアルゴリズムにより用いるソースを決定する。複数のノードが 1 つのソースを用いるときには、マルチキャストを計画する
2. 各ノードに到達するデータ量のスループットを最大化するよう、各データ転送に対して転送量を割り当てる。計算には線形計画法を用いる

3. 立てた転送計画をデータ転送ごとに再検討し、もし用いるソースを変更することでその転送のバンド幅を改善できる場合にはそのように変更する。変更後、再度線形計画法を用いて各転送の転送量を再計算する

以下、各フェーズについての詳細を説明する。

3.2 ソースの選択およびマルチキャストの計画

あるデータの転送について、大きなデータを一つのソースから多数のノードに転送するには、パイプラインを用いるのが効率的である。これは、ソースを先頭とし、転送先のノードを一列に並べ、順番に断片化したデータを送る。データが十分に大きい場合、全てのノードでデータはほぼ同時に到着する。また、パイプラインの転送速度は最も細いリンクに制約される。

実用上多くの場合、ネットワークトポロジは図1のような閉路を持たないツリー構造で表される。このツリーはノードとスイッチで構成されており、ノードは必ず一本のみのリンクを持つものとする。ここで最大のバンド幅を持つパイプラインは、各ノード・スイッチを幅優先でなぞってノードを並べることで求められる。ツリーの指定された全てのノードを接続する木は必ず全てのリンクを通るので、パイプラインの最大スループットは最小のバンド幅を持つリンクを超えることができない。ここで先ほど求められたパイプラインでは、各リンクが上り下り共に一回ずつ用いられるので、上り下りの転送が干渉しない場合、最小のバンド幅のリンクの値がそのままパイプラインのバンド幅になる。

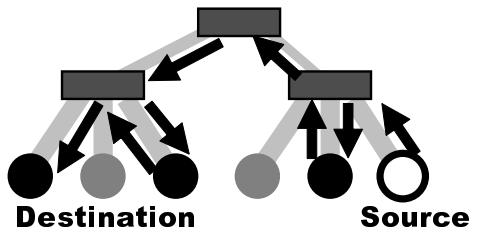

図 1. Pipeline multicast for large data

複数のノードがデータのコピーを持っていて複数のノードがそれを必要としているときは、ファイルを必要としている各ノードが最も近いソースを用いることで、細いリンクを用いる必要が無くなり、パイプラインの転送速度は向上する可能性がある。定性的には、ネットワーク的に遠いノードからファイルを転送した場合よりも、近隣のノードから転送した場合の方が早く転送が完了するということが言えるが、定量的なトポロジ・バンド幅情報を用いることでより正確な計画を立てることが出来る。

このようなパイプラインを計画する際、データを必要としている各ノードは最大のバンド幅で到達できるソースノードを探す必要がある。これは、グラフの最小木問題に対する既知のアルゴリズムである、クラスカルのアルゴリズム (Kruskal's Algorithm) を少し変形して用いると簡単に計画することができる。

まずは問題を定式化する。あるデータを多数のノード $\{dest_i \in D\}$ が必要としているとき、複数のノード $\{src_i \in S\}$ がそのデータのコピーを持っている。また、これらのノードとスイッチをつなぐリンク群 $\{l_i \in L\}$ がある。このとき、L の部分集合 L_a が存在し、L_a によって全ての $\{dest_i\}$ がそれぞれ一つの src_i に接続されているとする。リンク l のバンド幅を $bandwidth(l)$ と表すことにすると、$min(bandwidth(l_i)|l_i \in L_a)$ を最大にするような L_a を求めると、この L_a が求めるパイプラインを構成する枝である。

次にアルゴリズムを示す。A はソースにつながっているスイッチ/ノードを表し、B はデータを必要としているがまだソースにつながっていないノードを示している。L_a が最終的な出力となる部分木 (但し不要なリンクを含む場合がある) である。l が接続する二つのノードまたはスイッチを $e_0(l), e_1(l)$ と呼ぶことにする。

$A = S$
$B = D$
$C =$ (全ノード) $- S - D$
$L_a = \emptyset$
for descending order of bandwidth ($l \in L$) :
    if $e_0(l) \in A$ and $e_1(l) \in A$:
        continue (l を採用しないで次のループへ)
    else if $e_0(l) \in C$ or $e_1(l) \in C$:
        continue (l を採用しないで次のループへ)
    L_a にリンク l を追加
    if $e_0(l) \in A$:
        if $e_1(l) \in B$: B から $e_1(l)$ を除く
        else : ($e_1(l)$ はスイッチである)
            l の追加により A に接続されたノードを
            A に追加

if $B == \emptyset$: break
if $e_1(l) \in A$:
if $e_0(l) \in B$: B から $e_0(l)$ を除く
else : ($e_0(l)$ はスイッチである)
l の追加により A に接続されたノードを
A に追加
if $B == \emptyset$: break

トポロジ内のリンクをバンド幅順に並べ、順に追加していく。5 行目がそのリンクを L_a に採用する条件で、これを図 2 に詳しく示す。リンクが採用された場合、そのリンクを採用したことにより A から到達可能になったノード/スイッチを A に追加し、それが B に含まれていた場合 B から除く。B が空になるまでリンクを追加していくと、L_a として元々のトポロジのいくつかの部分木 (森) が得られる。この森に含まれるソースノードから順に幅優先でノードを並べると、求めるパイプライン群が得られる。

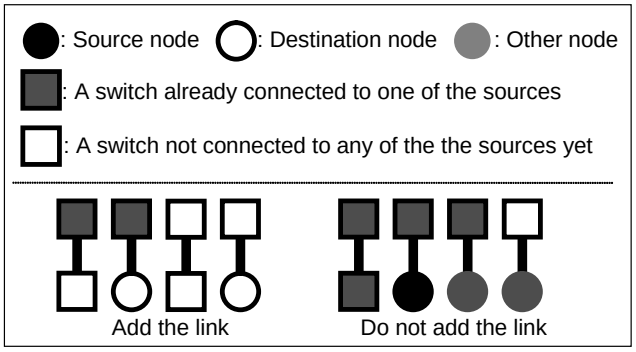

図 2. Kruskal's Algorithm (Add a link)

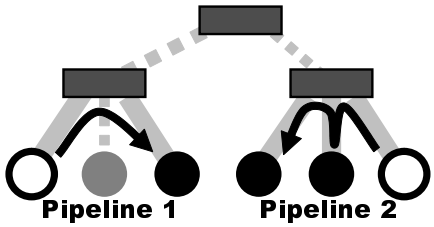

図 3. A forest obtained by Kruskal's Algorithm

このアルゴリズムのオーダーは、ノード数を n、最終的な木を構成するリンク数を m、ノード集合の探索・追加・削除が $O(\log n)$ で行えると仮定するとき、$m \log(n)$ となるので、トポロジが大規模になるとこのパイプライン作成のアルゴリズムも時間がかかってしまうことになる。これに対しては、ソースでもなくファイルの転送先でもないノードは絶対に転送に関わらないので、アルゴリズムの実行前に関係の無いノード・リンクを省くことで高速化が図れると思われる。

また、こうして得られた L_a は、条件を満たすあらゆるパイプライン群の中で、最小のリンクのバンド幅が最も大きくなっている。これを以下に示す。L_b が存在し、L_b 内の最小のバンド幅が L_a 内の最小のバンド幅よりも大きかったとする。ここで L_a を構成する際、アルゴリズムが停止直前にに最後に追加した枝 l_i は、データを必要としているあるノード $dest_i$ とスイッチを接続したリンクのはずである。この $dest_i$ は必ずパイプラインに接続されていなければならないので、L_b にも含まれているべきである。ここで l_i は L_a 中で最小のバンド幅の枝であるから、Lb 中で最小バンド幅の枝の値が L_a 中で最小バンド幅の枝の値より大きいという仮定に矛盾する。

3.3 線形計画法を用いた転送量の割り当て

データの転送計画が与えられたとき、各ノードに到着するスループットを最大化することを考える。転送は特に計画を行うことなく進めることも出来るが、トポロジ情報を元にして各転送が用いるバンド幅を明示的に計算し、指定して転送を行うことでスループットを大きく出来る場合がある。例えばあるリンクを二つの転送が共有している際、片方は多数のノードが関わるマルチキャストで、片方が 1 ノードのみへの転送である場合、前者により多くのバンド幅を与えることで、全体のスループットは向上する。実際にはある転送のバンド幅は複数の制約によって定められているので、これを定式化した上で線形計画法を用いて最適化を行う。このイメージを図 4 に示す。

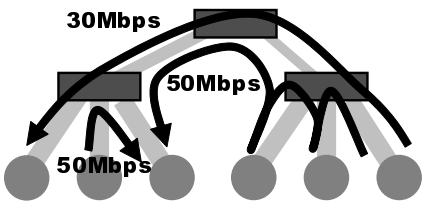

図 4. An Example of Bandwidth Assignments

整数 $\{1..l\}$ によって識別されるデータ転送が l 個あり、それぞれのバンド幅が $\{bw_i|1 \leq i \leq l\}$ であり、マルチキャストの到着ノードの数は $\{n_i|1 \leq i \leq l\}$ 個あるとする。このとき最適化したいのは、一定時間当たり各ノードに到着するデータ量であるから、$\sum_{k=0}^{l} bw_k * n_k$ と表すことができる。

ここで、バンド幅が c_i であるリンク L_i を通るデータ転送の集合を S_i とするとき、あるリンクを通過するデータの総量が c_i 以下であるという条件は、$\sum_{k \in S_i}^{l} bw_k \leq c_i$ と表すことができる。同様に全てのリンクについての条件を書き下し、線形計画法を用いると bw_l の値、あるいは関係が求まる。出力として値が定まらず等式が出力された場合は、これを満たすようなバンド幅を適当に定める。

スケジュラーに新しいタスクが動的に追加されるような環境では、特定のファイル転送にバンド幅が全く割り当てられず、そのタスクがずっと実行可能にならない、いわゆる飢餓問題が発生し得る。これを防ぐため、転送計画においては一定時間かかっても転送が開始されないファイルに対し一定のバンド幅を保証するような工夫が必要かもしれない。これについては、今後シミュレーションによっても確認したい。

3.4 パイプラインの再計画

こうして得られた転送計画は、各データ転送がリンクを共有しない場合にはスループットを最大にするような計画になっている。しかし、実際には各転送の間でリンクが共有されている可能性が高いので、複数のソースを持つデータ転送を対象に、初めに計画したマルチキャストのパイプラインを再検討する。このアルゴリズムを以下に示す。

あるデータの転送を再計画する手順は以下の通りである。まず全体のバンド幅トポロジから、各転送計画において使われたバンド幅を取り去ったバンド幅トポロジを作る。これに現在そのデータ転送に割り当てられているバンド幅を加え、このバンド幅トポロジを元にクラスカルのアルゴリズムを再度適用する。もし得られた結果が当初の結果と同じなら、元々の転送計画が最適である。もし異なる結果が得られた場合は、これを採用する。ツリーが変化するイメージを図 5 に示す。

この再計画の操作を全てのデータ転送について行い、もし転送計画が変化した場合は線形計画法によるバンド幅の計算を再度行う。

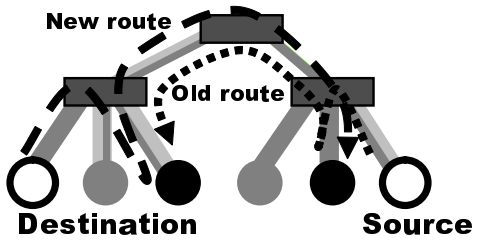

図 5. An Exmple of Transfer Re-planning

3.5 転送開始後のバンド幅の再計算

データ転送を開始した後、ネットワークの状態が変化した場合には、バンド幅の割り当てを変更する必要がある。例えばあるデータの転送が完了した場合や、外的な要因によりバンド幅が大きく変化した場合などが該当する。また新しいタスクのスケジューリングが始まった場合も、現在進行している転送のバンド幅を再計算する。

再計算の際に現在行われている転送のソースは変更しないが、割り当てるバンド幅は線形計画法を用いて再設定し、スループットの最大化を行う。動的にバンド幅を測定する方法としては、Nettimer[6] が用いているような、小さなパケットを多数用いた測定を利用したい。

4 進捗と評価方法

現在、本アルゴリズムによるスケジュラーの実装を進めている。まずは実際のデータ転送やバンド幅測定・タスク実行を除いたアルゴリズムのみを実装し、シミュレーションによりトポロジを考慮しないアルゴリズムとの比較を行う。次にシステムを実装し、実際の環境・タスクを用いた評価を行う。トポロジ情報及びバンド幅の測定は既存のライブラリを用いることを検討する。

5 おわりに

データ・インテンシブなアプリケーションに対応し、データ転送時間を最適化する分散タスクスケジュラーのアルゴリズムを提案した。本手法は、データ転送が少なくなるよ

うなタスクスケジュールを立てる点、与えられたタスクスケジュールに対し効率的なデータ転送を行う点の二点から構成されている。タスクのスケジュールは 2 章で述べた発見的手法により作成するが、その過程で各スケジュールに対し 3 章で提案したような手法でデータ転送の最適化を行う。ノードのネットワークトポロジ及び各リンクのバンド幅が与えられると仮定したとき、このアルゴリズムはマルチキャストの利用と線形計画法を用いて、複数のファイル転送が同時に行われるとき、各ノードに到着するデータのスループットを最大にする。

本アルゴリズムでは、あるデータを持つホストが複数存在する場合、トポロジ的にリンクを共有するような転送を避けるような計画が立てられる。また、リンクを共有する複数の転送内では、スループットを上げる上でより重要な転送により多くのバンド幅が用いられる。これにより、各ノードに分配されたタスクが必要とするデータがより早く転送され、より早く実行に移ることができる。

今後はスケジューリングアルゴリズム・スケジュラーの実装を進めるとともに、シミュレーションによるアルゴリズムの優位性も確認していきたい。

参考文献

[1] Jia Yu and Rajkumar Buyya “A Taxonomy of Scientific Workflow Systems for Grid computing” Special Issue on Scientific Workflows, SIGMOD Record, ACM Press, Volume 34, Number 3, Sept. 2005

[2] Tracy et al. “A Comparison Study of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems ” (TR-ECE 00-04)

[3] Mandal, A. et al. “Scheduling Strategies for Mapping Application Workflows onto the Grid. ” IEEE Symposium HPDC 14, 125.134. (2005)

[4] K. Cooper et al. “New Grid Scheduling and Rescheduling Methods in the GrADS Project” Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International (2004)

[5] 町田悠哉 滝澤真一朗 中田秀基 松岡聡 “レプリカ管理システムを利用したデータインテンシブアプリケーション向けスケジューリングシステム” ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS2006), 2006, 9-16. 4,

[6] Kevin Lai et al. “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay” SIGCOMM ’00, Stockholm, Sweden.

[7] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, “Condor-G: A Computation Management Agent for Multi-Institutional Grids.,” Cluster Computing, vol. 5, pp. 237-246, 2002.

[8] http://www.cs.wisc.edu/condor/dagman/

[9] Ewa Deelman1 et al. “Pegasus: Mapping Scientific Workflows onto the Grid” Lecture notes in computer science (Lect. notes comput. sci.) Grid computing (Nicosia, 28-30 January 2004, revised papers)

[10] Luiz Meyer “Planning spatial workflows to optimize grid performance” Proceedings of the 2006 ACM symposium on Applied computing Dijon, France (DSGC)

[11] Tracy et al. “A Comparison Study of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems ” (TR-ECE 00-04)

[12] Huadong et al. “Dynamic Co-Scheduling of Distributed Computation and Replication” Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID’06) - Volume 00 CCGRID ’06

[13] Phan et al. “Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm.” In Proceedings of the11th Workshop on Job Scheduling Strategies for Parallel Processing. Cambridge, MA. Springer-Verlag, Berlin, Germany 2005

[14] A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou, C. Pautasso, T. Heinis, R. Gronmo, Hjordis Hoff, Arne-Jorgen Berre, M. Glittum, S. Topouzidou “Developing scientific workflows from heterogeneous ser-

vices ” June 2006 ACM SIGMOD Record, Volume 35 Issue 2

発表文献

- 高橋慧, 田浦健次朗, 近山隆. マイグレーションを支援する分散集合オブジェクト. 並列／分散／協調処理に関するサマーワークショップ（SWoPP2005），武雄，2005 年 8 月.
- 高橋慧, 田浦健次朗, 近山隆. マイグレーションを支援する分散集合オブジェクト. 先進的計算基盤シンポジウム（SACSIS 2005），筑波，2005 年 5 月.