社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－HPC－116 (2)
2008／8／5

自動取得したネットワーク構成情報に基づくMPI集合通信

吉富翔太† 斎藤秀雄†
田浦健次朗† 近山隆†

頻繁に通信を行う大規模並列計算の性能を向上するためには，多くの計算機が一斉に通信を行う集合通信の最適化が必要不可欠である．またマルチクラスタ環境の普及により，ヘテロな環境で動作する集合通信の効率化が求められている．しかし多く既存の研究では予め手動でネットワークの情報を設定をしなければならず，巨大・仮想的な環境では大きな労力を必要としたり，ネットワークの細かい階層構造までは考慮はしていないというような問題点がある．

本稿ではネットワークトポロジーを推定するアルゴリズムを組み込んでアプリケーションの実行直前に自動的に取得したトポロジーの情報利用して集合通信を効率最適化する方法を提案する．我々の手法では得られたネットワークトポロジーと計算機間の遅延の情報を元に，ボトルネックとなるリンクやスイッチのパケットロスを防止することなどを考慮して，効率の良い通信スケジューリングを行うことが特徴である．そして7クラスタ230ノードの環境でBroadcastとGatherの実装と評価を行い，既存の手法に比べて最大で45 %の性能向上を実現した．

An Efficient MPI Collective Communication Algorithm Using Inferred Network Information

Shota Yoshitomi,† Hideo Saito,† Kenjiro Taura† and Takashi Chikayama†

Optimization of collective communications is the key to achieving high performance parallel computing. Also design of the efficient collective communicaton on heterogeneous network-based systems plays an important role in the reduction of the computing time. In previous works, however, it is required to supply network information which is a troublesome task, and also these works are often not well suited for the hierarchical and complex network.

In this paper, we propose an efficient collective communication framework using the automatically inferred network information. Our proposal develops adaptive network-aware communication models with the link latency and the network of the topology as a tree. We implemented this system on the MPI, and confirmed that broadcasts and gathers execute faster than a topology-unaware implementation on 230 hosts in 7 clusters.

1. はじめに

近年のGrid環境の普及により，大規模な科学技術計算をGridを用いて並列実行することが積極的に行われるようになってきた．このような環境において大規模な計算を実行するときに，計算機間で頻繁に通信を行うような計算処理では，処理性能の向上の為には通信時間の短縮が求められる．特に，複数の計算機間で一斉に多くの通信を行う集合通信の性能が全体の計算処理時間に大きく関わってくる．従ってGridでの通信を伴うアプリケーションの性能を向上するためには，アプリケーションに用いられる集合通信の性能を向上させ，効率よく通信することが必要不可欠である．そこで，本研究では特にMPI[1]の集合通信に着目し，その性能効率向上法について論じる．

MPIのアプリケーションで用いられる集合通信に関しては，従来よりノードの局所性を考慮してデータを通信する順序やリンク上を流れるデータ量を制御したり，頻繁なデータのやり取りはできるだけ比較的遅延の少ないクラスタ内で行ってクラスタ間などの遅延の大きい部分での通信をなるべく抑制するといったような，集合通信の最適化に関する様々な研究が行われてきた．

しかし，これらの研究の多くが個々の処理について細かい設定を手動で与えるものである．例えばGrid環境で動作するMPIの集合通信に関する多くの研究[4),5)]では，MPIのシステムが用いるノード間の局所性が

† 東京大学
University of Tokyo

分かっているということを仮定している．そして，使用するノードのホスト名やIPアドレスといったノードの情報をあらかじめ手動で設定を行っていなければならない．

そのためこのシステムは，全てのユーザが同じノードの集合のみを用いて，さらに使えるノード集合が滅多に変更されないような静的な環境でしか用いることができない．管理者が異なる複数のノード群（例えば異なる管理者が管理するマルチクラスタ環境）ではネットワークの情報を統一的に設定するのは困難であるし，ノード数が数千，数万と増えるとその手間は爆発的に増大する．またVLANのように仮想化された環境や，使える計算ノードの集合が頻繁に変更される環境では設定の更新に大きな労力が必要となる．

一方で，このような使えるノード集合が頻繁に変更されるようなよりオープンで動的な環境における，ネットワークを考慮した並列計算を支援するツールの研究も多く行われてきた．例えば，白井らによる研究[3]では，ネットワークに負荷をかけずに，非常に短時間でネットワークのトポロジーを推定する手法を実現している．

そこで本発表では，予めこの手法を用いてアプリケーション実行直前に手動設定に頼らず実際に使用するトポロジーや遅延といったネットワークの構成情報を自動的に取得し，その情報に基づいて，ネットワークの階層構造を考慮しボトルネックリンクを減らしたり，輻輳をなるべく回避する集合通信のスケジューリングを考える．

2. Gridに適応した既存のMPI集合通信

ここではGrid環境に適応する既存の集合通信の特徴と問題点について述べる．

Thilo Kielmannらの MagPIe[4] は，クラスタ内とクラスタ間での通信のアルゴリズムを使い分けることにより WAN のように遅延の大きなノード間の通信を極力減らし，なおかつクラスタ間のバンド幅を無駄遣いしないという特徴を持っている．

ところがMagPIeはネットワークの種類をLAN,WANという遅延の大小のみを基準とした2種類に限定し，メッセージ長さの短い集合通信のみを考慮した実装である．またLAN,WANの判別を行うために，ネットワークトポロジーや遅延といった情報を事前に設定しなければならない．その点で本研究が目的とするユーザによる手動でのネットワーク関係の事前の設定が不要であり，ネットワークの階層構造まで意識した集合通信とは異なっている．

またSathish S. Vadhiyarらによる集合通信をシステムに応じて自動的にチューニングする手法[5]では，実際のシステム上で実行ノード数やデータサイズを変化させて一連の集合通信操作のアルゴリズムの性能を試す試験を行い，各ノード数やデータサイズに対して最短時間で終了したアルゴリズムを次回以降利用するように集合通信をそれぞれのネットワーク用に自動チューニングする方法を提唱している．問題点としては，何人ものユーザが同時に計算処理を行うような環境では，他ユーザの処理を邪魔せずかつ自分のアプリケーションのパフォーマンスを低下させないために，アプリケーションの実行直前にその時点で空いているノードを選択し計算処理を行うことがある．そういった実行環境が動的に変化しうる場合には予め試験を行うことができないということが挙げられる．

3. 提案手法のモデル

3.1 概　　略

一方我々の提案するネットワーク情報を用いた集合通信の概略は以下の通りである．

(1) 3.2の手法を利用して，アプリケーション実行直前にネットワークの構成情報を自動的に取得する．このとき取得される情報は，実際にMPIのアプリケーションを実行するノードの全体集合と同一のものである．

(2) 得られたデータを全てノードのメモリ上に配置し，全ノードが等しくそのデータを参照することができるようにする．

(3) 集合通信を行うAPIが呼び出されると，1で得た情報に基づき，各々のノードがローカルに同一のポリシーに基づいてスケジューリングを行う．

(4) 実際のノード間通信を行う．

また，実際に手法を設計実装するにあたり次の前提を置く．

- 集合通信は全ノードで同時に開始されると仮定する．
- 3.2の手法で取得できる情報は，ノード，スイッチの接続関係と遅延の2つである．バンド幅に関する情報は得られない．従って，バンド幅についてはいくらかの見積りを行うものとする．

そのもとで，集合通信の全実行時間の短縮を図る．

3.2 高速なトポロジー推定手法

次に本発表において利用する，白井らによるネットワークトポロジーを高速に推定する手法[3]について，その特徴を述べる．

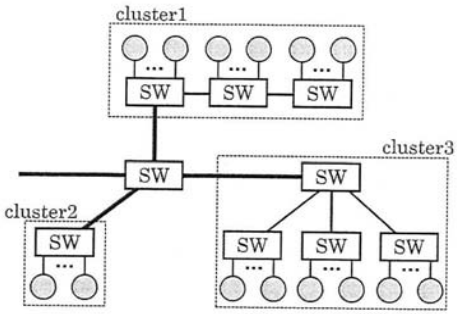

図 1　推定されたネットワークトポロジーの一例

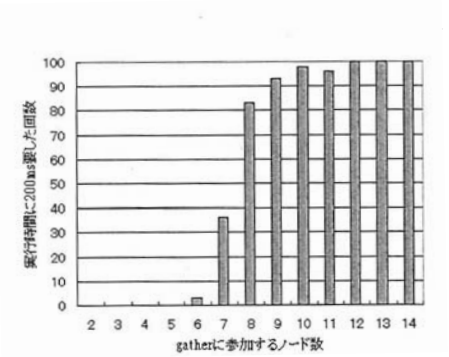

図 2　単一スイッチに接続された 2〜14 ノードの gather

この方法ではノード間の RTT の測定をユーザが手動の設定を一切必要とせず，ヘテロな環境で高速かつ低負荷にネットワークのトポロジーを推定する．推定できるトポロジーの例を図 1 に示す．この図のように，スイッチとノードのツリー状の論理的な接続関係を求めることができる．また，各リンクの遅延も合わせて取得可能である．

3.3　トポロジーを利用した集合通信

集合通信において，前述の手法により自動取得したネットワークトポロジーを利用することによる利点は，以下の 3 つが考えられる．

3.3.1　ネットワークに関する設定

まず，既に述べたように非常に大規模・仮想的な環境であっても，トポロジーを自動取得できるので，集合通信に関してネットワーク情報の設定を行わなくても良いことが挙げられる．

3.3.2　ボトルネックリンクの検出

次に，ネットワークにおいて途中のリンクを複数の計算機が共用している場合は，そのリンクがボトルネックとなる．一番顕著な例はクラスタとクラスタを繋いでいるリンクであるが，クラスタ内に限っても，複数のノードが接続されているスイッチ同士を繋ぐリンクは集合通信においてボトルネックとなる．このようなボトルネックリンクを考慮して通信をスケジューリングするのは重要である．

3.3.3　ネットワーク中のスイッチやルータの検出

ネットワーク中のスイッチの位置を把握することは以下の点から非常に重要である．gather や reduce，alltoall といった，ある 1 ノードに他のノードからデータが集中する操作においては，ネットワークの輻輳やスイッチでのパケットロスの影響により，期待された性能が出ないことがある．

例えば，我々が知る限りでは，同一の Gigabit Ethernet スイッチ (DELL PowerConnect5234) に接続されているノードを数台用いて gather を行った場合，通信スケジューリングを考慮せずに全ノードが一斉にデータの送信を行うと，データサイズとリンクのバンド幅や遅延の値から算出される理想的な gather 実行時間よりも 200ms 以上余分に時間がかかるという結果が観察された．図 2 にその結果の一部を示す．これは，各ノードが 10KB ずつ残りの 1 ノードに対して送信する gather を 100 回繰り返すという操作を，gather に参加するノードを 2〜14 ノードまで変化させて行ったときに，実行時間に 200ms 以上要した回数の度数分布である．この場合の理論的な gather の実行時間は多くとも 20ms 程度であったが，gather に参加するノードが増えるにつれ，200ms 以上の実行時間を要する頻度が増加している．

この現象の原因はスイッチでのパケットロスに起因するものである．ある 1 ノードに対して同時に複数のノードから送受信を行うと，途中に存在するスイッチのパケットバッファ容量が十分でない場合にはバッファがあふれてしまい，パケットロスが生じる．TCP はパケットロスが起きた場合に高速な再送を行う機構を実装してはいるが，スロースタート期間中のように cwnd が小さい場合にはその機構が十分に働かずに送信側がタイムアウト (RTO) するまで待機してから，自発的にパケットを再送する．実行した Linux kernel 2.6.18 の環境では，RTO の時間が 200ms に設定されており，図 2 の結果において実行時間が 200ms を超えているものにはこのタイムアウトの現象が確認された．MPI のようにバースト的に通信を多く行うアプリケーションではスロースタートが頻発し，結果として以上の現象が起きやすくなると考えられる．

提案手法の場合，推定されたネットワークのトポロジーを一つの情報として利用することで，このようなパケットロスとそれに伴うタイムアウトを引き起こす一つの要因であるクラスタ内のスイッチをあらかじめ検出することが可能となる．そしてそれぞれのスイッチについて，宛先ノードが同一なデータが複数のノー

ドから同時に集中しないように適切なスケジューリングをすることができる．

4. 設計及び実装

本研究では，提案手法の集合通信モデルを Grid 環境に対応した MPI のライブラリである，MC-MPI[2] 上に C 言語を用いて実装する．そして単一クラスタ内での集合通信に対応するほか，複数のクラスタをまたいだ Grid 環境での集合通信も想定する．ただし，今現在 NAT や Firewall が存在する環境ではネットワークトポロジーの情報が取得できないので，今のところグローバル IP を持つノードのみを考慮するものとする．

4.1 ネットワーク構成情報の取得

MPI アプリケーション実行直前に 3.2 の手法に基づくネットワークトポロジー推定プログラムを実行し，ネットワーク構成情報を取得する．このプログラムの実行に伴い，(1) ノードとノードの間の遅延の情報，(2) ノードとスイッチの接続関係の 2 つのデータを全ノードが保有することができる．

4.2 集合通信のアルゴリズム

続いて，集合通信のツリー構造の設計やスケジューリング手法について集合通信操作を一対全通信として (Broadcast, Scatter) 及び (Gather, Reduce) が全対全通信として (Alltoall, Allgather, Allreduce) の 3 つに大別して述べる．

4.2.1 Broadcast, Scatter

Broadcast や Scatter は root が持つデータを，全ノードに対して送信する集合通信である．

α をノード間の遅延，β をノード間のリンクのバンド幅，M を送信するデータの総量，O_s,O_r をそれぞれ送信，受信の際のオーバヘッドとする．すると，2 ノード間のデータ送受信にかかる時間 T_0 は

$$T_0 = O_s + \alpha + \frac{M}{\beta} + O_r \qquad (1)$$

である．

一方，Broadcast や Scatter など，単一の root からデータを他のノードに送信するような場合，何らかのツリー構造に沿ってデータの転送が行われる場合が多い．典型的な例として，ノードの総数 n に対して，(a) 深さが $O(1)$ の Flat tree，(b) 深さが $O(logn)$ の Binomial tree，(c) 深さが $O(n)$ の Chain などのアルゴリズムが知られている．これらのツリーのアルゴリズムにおいては，ネットワークの遅延の大きさやバンド幅，データサイズとの比率により最適なアルゴリズムが異なる．具体的には，高遅延の環境では Flat tree の方が Binomial tree に比べて性能が良い．反対に，

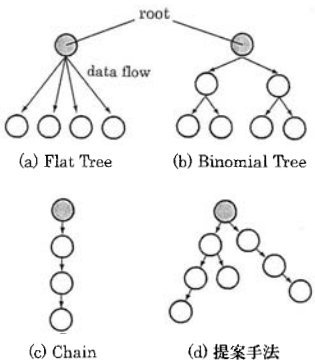

図 3 Broadcast における様々なツリー構造

データ送信のオーバーヘッド時間 O_s に比べて遅延 α が小さい場合は，Binomial tree や chain アルゴリズムの方が効率的となる．Grid 環境のように様々な遅延のリンクが存在するような環境では，ある特定のツリーを構築するのではなく，トポロジーの形に合わせてスケジューリングすることが必要となる．また，パイプライン状にデータを順次転送をすることも効率を上げる上では有効な手段となる．

提案手法の Broadcast は以下の方針で root からのツリーを構築していく．O_s, O_r, β については，ある程度の見積りをした値を用いる．

(1) 既にデータを所有しているノードの集合 S を定義する．明らかにこの段階では S の要素は root のみである．また，通信開始からの経過時間を見積もるため，経過時間 $T_i (0 \leq i < p, p =$ 全ノード数) も定義する．

(2) S に属さないノードを一つ選び，S に属する全てのノードに対し，$T_j = T_i + \alpha_{ij} + n\beta_{ij} (i \in S, j \in \bar{S}$ を最小にする i, j の組を求める．この組が実際に通信を行う組となる．

(3) j を S に加える．

(4) その i, j に対し，$T_j = T_i + O_r + \alpha_{ij} + n\beta_{ij}, T_i = T_i + O_s$ として T_i, T_j を更新する．

(5) 全てのノードが S に属するまで 2〜4 を繰り返す．

このスケジューリングアルゴリズムに従って作成される通信のツリー構造は図 3 (d) のようになる．

4.2.2 Gather, Reduce

Gather や Reduce は root ノードに対して全ノードが保有しているデータを送信し，root ノードがそれらのデータを集約するというものである．

これらの，1 ノードにデータを集めるような集合通信は，前述したように途中のスイッチでのパケットロ

スをなるべく防止する為に，通信をスケジューリングすることがスループットを低下させない上で重要である．提案手法では，アプリケーションのレベルでデータの送受信開始時刻を制御することにより，同一の宛先へのデータが一つのスイッチに集中することを防ぐものとする．そこで，まず以下の条件を課す．

- いかなるノードも，2つ以上のノードからのデータを同時に受信することはない．
- ネットワーク中の任意のスイッチ S_i に対して，同一の宛先へと向かう複数のストリームが同時に集中しないようにする．(A)

この条件を満たすように次の手順でデータを送信するノードをスケジューリングする．

(1) 初期値として，各ノードは時間 $t_s = 0$ で送信を開始すると仮定する．

(2) root に一番近いスイッチに対して，(1) 式を用いて各ノードからの到達時間を見積り，条件 (A) を満たすように重複しているノードの送信開始時間 t_S を重複しないよう sleep することで後ろにずらす．基本的に root との遅延が大きいノードから優先的に後ろにずらすようにする．

(3) 末端のスイッチに向かって順に同様に，条件 (A) を満たすように処理を行う．

この場合のスケジューリングに要する計算量は，高々 $O(n^2)$ である．各ノードはこれらの手順に操作に従って定められた送信開始時刻にデータの送信を開始する．

4.2.3 Alltoall, Allgather, Allreduce

続いて，全対全通信に関して，アルゴリズムの選択方針について述べる．全対全通信は Allgather, Allreduce, Alltoall の 3 つ取り上げる．これらの通信は送信するデータの種類やサイズが異なるだけで，本質的には同等のものと考えることが可能である．全対全通信を行う集合通信は，一対全の通信とは異なり，複数のノードが同時に異なるデータの送受信を行うために，一対全通信のように root を頂点とするツリー構造を作るだけでは単純に全ノードの通信に拡張するのは困難である．また，全対全通信を効率良く行うための様々なアルゴリズムも提案されているが，実行ノードの数に制限が加えられていたり，ヘテロな環境では適用し難いことなどが問題となる．

より一般的には，ネットワークは全二重であり任意のノードは一つずつのノード達との送信・受信が同時に行えると仮定すると，これらの全対全通信は Broadcast,Scatter と Gather,Reduce を組み合わせて通信をスケジュールすることにより実現できると捉えることが可能である．ノード P_i から P_j への通信を，ノード P_j に対する Job と捉えれば，全対全通信は Open Shop Scheduling 問題に帰着できる[7]．この Shop Scheduling 問題は NP 完全であるが，ある種の近似的な解法を用いることにより，多項式時間で解くことが可能となる[8]．

提案手法としては，この Open Shop Scheduling 問題に Gather と同様，

- パケットロスを防止するため任意のスイッチ S_i に同一宛先へのストリームが集中しない．
- 同一リンクに複数のストリームが集中した場合には，そのリンクの利用可能なバンド幅が減少する．そのため，任意のリンク L_i にストリームが集中しない．

これらの制約を課すことにより，全対全通信のスケジューリングを行うことを考える．

表 1 実験環境の構成: 7 クラスタ 230 ノード

ノード数	87	105	38
CPU	1.8GHz	2.33GHz	2.33GHz
Memory	1GB	4GB	8GB
OS(Linux)	Kernel 2.6	Kernel 2.6	Kernel 2.6

5. 性能評価

本提案手法の実装を行い評価実験を行った．本提案手法における各種集合通信を実装し，それぞれの集合通信について，通信開始から全ノードが通信を終えるまでの時間を測定するとともに，比較実験として同様の条件で既存の集合通信との性能の比較を行った．実験環境は，InTrigger プラットフォーム[6] を利用し，その中の 7 クラスタ，合計 230 ノードを用いて行った．実験環境の構成は表 1 に示す通りである．

5.1 Broadcast

提案手法における Broadcast を実装し，性能を測定した．また大きく分けてクラスタ内では Binomial tree, クラスタ外では Flat tree を用いる既存の詳細なトポロジーまで考慮しない手法との比較を行った．

提案手法の実行時間を 1 として正規化を行った結果を図 4 に示す．送信データ長が 1MB の段階で提案手法は比較手法に比べて 45 %の性能改善が見られた．これは提案手法がパイプライン転送による効率化と，ボトルネックリンクを減らすようなスケジューリングができているためである．

5.2 Gather

次に Gather について，様々なデータ長で性能測定した結果を図 5 に示す．比較対象として，topology-unaware な方法として全ノードがスケジューリングを

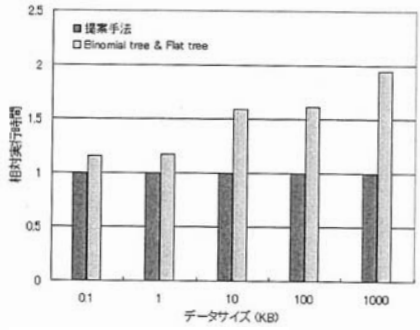

図 4 7 クラスタ 230 ノードにおける Broadcast

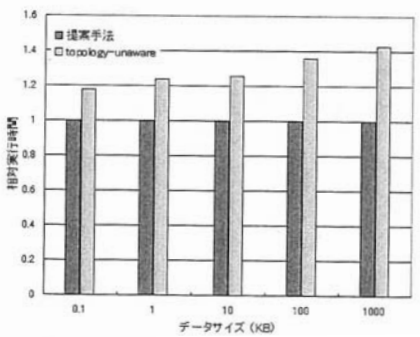

図 5 7 クラスタ 230 ノードにおける Gather

まったく考慮せず，一斉にデータを送信する方法を用いている．するとデータ長が 1MB の所で topology-unaware な方法と比べて 30 %程の性能が向上が見られた．これは，3.3 にあるようなスループットの減少が比較的避けられている為である．

6. おわりに

本論文では，あらかじめアプリケーション実行直前に自動的に推定・取得したネットワークの構成情報を利用することで，ユーザがネットワークの構造を意識し手動でネットワークに関する設定をする手間を取らせることなく，ネットワークの構造にを考慮した効率の良い様々な集合通信が行えるようなシステムを提案し，設計と実装を行った．また，実際に 7 クラスタ 230 ノードの環境で Broadcast と Gather について性能の比較を行い，従来の集合通信のアルゴリズムよりも集合通信の性能が最大で 45 %向上していることを確認した．

今後の課題として，まず全対全通信について提案手法の評価と性能の検討を行うことが挙げられる．また，本手法に用いたトポロジーの推定手法では，NAT や Firewall が一部存在する環境ではトポロジーの推定が行えないため，それらの存在する環境下でアルゴリズムを動作させることができなかったので，この点を改良する必要がある．

さらに今回は個々の集合通信が開始してから終了するまでの時間のみを性能の比較対象として実験を行った．しかし，ごく少ないデータ量の集合通信しか行わない場合には，トポロジーのデータを推定・取得しツリーを構築するためのオーバーヘッドが通信時間に比べて相対的に大きくなり，アプリケーションの実行時間が逆に増大してしまうことも十分考えられる．そのためトポロジーのデータをより素早く取得し，迅速に通信スケジューリングが行えるようにする必要がある．

謝辞 本研究の一部は文部科学省科学研究費補助金特定領域研究「情報爆発に対応する新 IT 基盤研究プラットフォームの構築」の助成を得て行われた．

参考文献

1) The Message Passing Interface(MPI). http://www-unix.mcs.anl.gov/mpi/ .
2) Hideo Saito, Kenjiro Taura.: Locality-aware Connection Management and Rank Assignment for Wide-area MPI. CCGrid 2007, pp. 249-256, 2007.
3) Tatsuya Shirai, Hideo Saito and Kenjiro Taura. A Fast Topology Inference — A building block for network-aware parallel computing. In Proceedings of the 16th IEEE International Symposium on HPDC, pp.11-21, Monterey, June 2007.
4) Thilo Kielmann, Rutger F.H.Hofman, Henri E.Bal, Aske Plaat, Raoul A.F.Bhoedjang. MagPie: MPI's Collective Communication Operations for Clusterd Wide Area Systems. PPoPP'99, May 1999.
5) Sathish S. Vadhiyar, Graham E. Fagg and jack Dngarra. Automatically Tuned Collective Communications. Supercomputing, ACM/IEEE 2000 Conference Publication Date: 04-10 Nov. 2000 page(s): 3- 3
6) Intrigger プラットフォーム: https://www.logos.ic.i.u-tokyo.ac.jp/intrigger/
7) Prashanth B.Bhat, Viktor K.Prasanna and C.S.Raghavendra. Adaptive Communication Algorithms for Distributed Heterogeneous Systems. the 7th IEEE International Symposium on HPDC, 1998.
8) H.Brasel, T.Tautenhahn and F.Werner. Constructive heuristic algorithms for the open shop problem. Computing, 51:95-110, 1993.