2007－HPC－111 (41)
2007／8／3

InTrigger：柔軟な構成変化を考慮した多拠点に渡る分散計算機環境

斎藤秀雄[†,††] 鴨志田良和[†] 澤井省吾[†]
弘中健[†] 高橋慧[†] 関谷岳史[†]
頓楠[†] 柴田剛志[†]
横山大作[†] 田浦健次朗[†]

大規模分散計算機環境 InTrigger について説明する．本プロジェクトでは国内の様々な教育・研究機関にクラスタを設置し，20-30 拠点 1000 コア以上の環境を構築することを目指しており，現在 6 拠点 514 コアが導入済みである．PXE ブートや IPMI などの技術を用いて多拠点に渡る分散計算機環境の構成を柔軟に変化させながら管理する方法を説明し，分散環境を使いこなすために本環境で提供されているツールをいくつか紹介する．また，InTrigger のような環境を構築することによって初めて行える実験があるということを示し，そのような実験を行うことによって得られた知見について述べる．

InTrigger: A Multi-Site Distributed Computing Environment Supporting Flexible Configuration Changes

Hideo Saito,[†] Yoshikazu Kamoshida,[†] Shogo Sawai,[†]
Ken Hironaka,[†] Kei Takahashi,[†] Takeshi Sekiya,[†]
Nan Dun,[†] Takeshi Shibata,[†] Daisaku Yokoyama[†]
and Kenjiro Taura[†]

We describe InTrigger, a large-scale distributed computing environment. Our goal is to install over 1000 cores spread out across 20 to 30 sites, and we have already installed 514 cores spread out across 6 sites. We explain how we use technologies such as PXE boot and IPMI in order to manage such a multi-site distributed computing environment while making flexible configuration changes. We also introduce a few of the tools that we provide to make it easier to use distributed environments. Finally, we show that some experiments can only be performed with an environment like InTrigger, and describe what we have learned by performing one such experiment.

1. はじめに

本稿では，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」で構築している大規模分散計算機環境 InTrigger について説明する．本プロジェクトでは国内の様々な教育・研究機関にクラスタを設置し，6 年間で 20-30 拠点 1000 コア以上の環境を構築することを目指している．

分散計算機環境を構築する試みは他国でも行われている．例えば，フランスには Grid'5000[1] があり，オランダには DAS-3[2] がある．大規模な分散環境を構築するという点では InTrigger もこれらの試みと似ているが，拠点数が Grid'5000 や DAS-3 より多いという点が特徴的である．

以降，まず 2 章で InTrigger 環境の概要を説明する．次に，3 章で InTrigger のような多拠点に渡る分散計算機環境の構成を柔軟に変化させながら管理する方法について説明する．その後，4 章で分散環境を使いこなすために本環境で提供されているツールをいくつか紹介し，5 章で多数のユーザが計算資源を共用するための利用ルールについて述べる．また，6 章では InTrigger のような大規模広域環境を構築して初めて行える実験があるということを示し，そのような実験を行うことによって得られた知見について述べる．最後に，7 章でまとめと今後の展望を述べる．

† 東京大学
The University of Tokyo
†† 日本学術振興会特別研究員
Research Fellow for the Japan Society for the Promotion of Science

表 1　各クラスタの仕様
Table 1　Specifications of each cluster

サイト名	設置機関	CPU	ノード数 (コア数)	メモリ	ディスク (ローカル)	ディスク (共有)	ネットワーク
chiba	国立情報学研究所	Pentium M 1.86GHz Core2 Duo 2.13GHz	70 (70) 58 (116)	1GB 4GB	300GB 500GB	9TB	グローバル
hongo	東京大学	Pentium M 1.86GHz Core2 Duo 2.13GHz	70 (70) 14 (28)	1GB 4GB	70GB 500GB	2TB	グローバル
imade	京都大学	Core2 Duo 2.13GHz	30 (60)	4GB	500GB	9TB	プライベート
kyoto	京都大学	Core2 Duo 2.13GHz	35 (70)	4GB	500GB	9TB	プライベート
okubo	早稲田大学	Core2 Duo 2.13GHz	14 (28)	4GB	500GB	9TB	グローバル
suzuk	東京工業大学	Core2 Duo 2.13GHz	36 (72)	4GB	500GB	9TB	グローバル
合計			327 (514)	888GB	119TB	47TB	

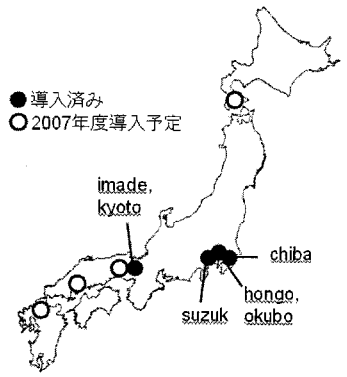

図 1　導入済み及び 2007 年度導入予定のクラスタの位置
Fig. 1　Locations of the clusters already installed and to be installed in Fiscal Year 2007

表 2　ping で測定したクラスタ間の RTT(ms)
Table 2　The inter-cluster RTT measured using ping (ms)

	chiba	hongo	imade, kyoto	okubo	suzuk
chiba	-	6.2	18	6.3	8.0
hongo	-	-	12	2.7	1.6
imade, kyoto	-	-	-	11	12
okubo	-	-	-	-	4.3
suzuk	-	-	-	-	-

2. InTrigger 環境の概要

本プロジェクトでは 6 年間で 20-30 拠点 1000 コア以上の環境を構築する予定であるが，現在は 3 年目に入ったところで，6 拠点 514 コアが導入済みである．**図 1** にこれまでに導入済みのクラスタ及び今年度導入予定のクラスタの位置を示す．また，**表 1** に各クラスタの仕様を示す．ネットワークの欄が「グローバル」となっているクラスタは全ノードがグローバル IP を持っている．一方，「プライベート」となっているクラスタはファイルサーバ及びいくつかの計算ノードのみグローバル IP を持っており，残りの計算ノードはプライベート IP しか持っていない．

表 3　iperf で測定したクラスタ間のバンド幅 (Mbps)
Table 3　The inter-cluster bandwidth measured using iperf (Mbps)

	chiba	hongo	imade, kyoto	okubo	suzuk
chiba	-	630	93	83	790
hongo	780	-	94	83	820
imade, kyoto	33	46	-	26	51
okubo	81	92	42	-	79
suzuk	66	580	95	78	-

表 2 にクラスタ間の RTT (Round-Trip Time) を示す．現時点では okubo と imade・kyoto の間の RTT が 18ms で最も長いが，今年度北海道と九州にクラスタを導入した後はさらに広域な環境になる予定である．また，**表 3** にクラスタ間のバンド幅を示す．全拠点が SINET[3] 経由で 1Gbps 以上の回線で接続されているので，測定されたバンド幅が数十 Mbps しかなかったところに関してはネットワークの設定を確認する必要がある．

3. 柔軟な構成変化を考慮した管理

様々な拠点に分散したクラスタ計算機の管理を行うとき，効率的に作業を行うためにはなるべく多くの作業をネットワーク経由で遠隔地から行うことができる必要がある．また，システムソフトウェアの研究では様々なソフトウェアをインストールしたり OS の設定を書き換えるような実験を行ったりする場合があるが，このようなとき，ソフトウェアパッケージをインストールするのと同じような感覚で気軽にクラスタ内の各ノードの OS を再インストールできると便利である．

InTrigger 環境ではこれらのことを実現するために，ネットワーク経由で OS のインストールを行うためのツールである Lucie[4] を拡張した．Lucie は Debian の FAI (Fully Automatic Installation)[5] を基に作成された自動インストーラであり，下記の手順でインストールを行う．

(1) ノードが PXE ブートにより起動
(2) インストール用サーバからインストール用カーネルをダンウロード
(3) ルートファイルシステムを NFS マウント
(4) イントールの実行

元の Lucie ではすべてのノードのインストールが完了するまで待機して，自動インストールを行う設定を解除した後に，各ノードの端末を操作して再起動を指示することが必要であるが，これでは遠隔インストールには向かない．そこで我々は，ステップ (4) の後に次のようなステップを追加することによってインストールが完全に自動的に行われるようにした．

(5) インストール用サーバにインストールの完了を通知
(6) 再起動

インストール用もしくはインストールされるカーネルなどに問題があると自動インストールが途中で止まってしまうが，そのような場合には IPMI (Intelligent Platform Management Interface) でトラブルシュートを行う．IPMI を用いるとノードの電源をリセットすることができ，また，IPMI の SOL (Serial on LAN) を用いるとコンソール表示を遠隔ノードにリダイレクトすることができる．

また，元の Lucie はステップ (4) で OS だけではなくすべてのソフトウェアをインストールするが，ルートファイルシステムを NFS マウントした状態でのトラブルシュートは困難である．そこで我々は，必要最低限のソフトウェア以外はステップ (6) の後，ノードがローカルディスクから起動した状態でインストールするようにした．再起動後にインストールするソフトウェアの設定はスクリプトで行い，スクリプトを Subversion で管理することによって全ノードに共通のソフトウェアをインストールする．具体的には，Subversion にスクリプトをコミットしておくとノードの再インストール時に自動的にそれがチェックアウトされ，実行される．

4. 提供されているツール

本章では，分散環境を使いこなすために InTrigger で提供されているツールをいくつか紹介する．従来から並列分散環境を対象としたツール，ミドルウェアは数多く存在する．しかし，その多くは単一クラスタ環境を対象としている．我々が構築した InTrigger は，既に 6 拠点ものクラスタをまたがり，今後も増加していくことを考慮すると，新たな広域分散環境に対応したツール，ミドルウェアが必要である．

4.1 GXP

分散環境を効率的に扱うためには多くの計算機への素早いコマンド投入やこれらの計算機を協調動作させる分散アプリケーションを簡単に記述できることが重要になる．また，これらを複雑な設定を行わずに可能にすることが求められる．Grid eXPlorer(GXP)[6] はコマンドラインインタフェースの並列シェルで，複数拠点にまたがる分散環境で効率的に利用できるログイン機能とコマンド実行機能が実装されている．

ログイン機能は，少ない設定で現在利用可能なノード群に素早くログインが可能になるように設計されている．あらかじめ各ノードへのログイン方法を指定しておき，ログインコマンドを実行することで指定した計算機に並列にログインを行い，GXP を終了するまでセッションを維持する．直接ログインできない計算機へも，一旦ゲートウェイのノードにログインした後にその内部のノードにログインを行うように設定できるため，プライベートネットワークやファイヤーウォールがある環境でも使用することができる．ログイン方法の指定は，複数の可能なログイン経路を冗長に記述することができ，故障ノードが存在するなどしてログインできなかった場合は，自動的に他の経路を試す．ssh を使ったログインの他に，TORQUE[10] や Sun Grid Engine を経由してログインすることも可能である．また，インストールを簡単にするため，GXP はログインが成功したノードに対して，自分自身をコピーする機能を有している．

GXP では，コマンドを送信すると，それがすべてのノードで実行されるという単純なモデルでコマンドが実行できる．各ノードで実行されるコマンドへのデータのブロードキャストやコマンドの出力の集約を Unix シェルの入出力のリダイレクトのように記述することができるため，マスター・ワーカー型の並列プログラムを簡単に作ることができる．前回実行したコマンドの終了ステータスによって，次にコマンドを実行するときのノードを選ぶ機能を使うことで，特定のノードのみでコマンドを実行することも可能である．

4.2 VGXP

並列計算環境は，計算機の台数が増えるほど，また分散している拠点の数が増えるほど安定した状態で利

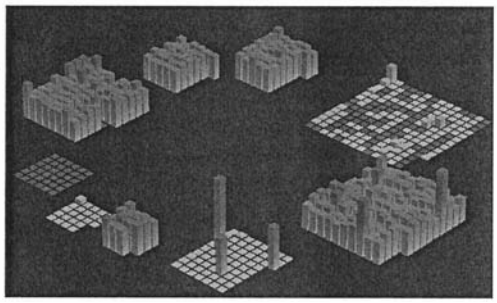

図 2 VGXP のスクリーンショット
Fig. 2 A screenshot of VGXP

用することが困難になる．

不測の問題が発生したときに，現在の計算環境の状態を迅速に把握して対処を行うためにはモニタリングシステムの存在が不可欠となる．

複数の拠点に対応できるモニタリングシステムとして，たとえば Ganglia や Nagios が存在する．これらのツールはノードの生死や監視項目の値があらかじめ設定した範囲にあるかどうかを判定してそれらを表示することができる．しかし，望ましい値の範囲を静的に決定することは難しいし，設定も煩雑である．また，これらのツールは 1 分から数分ごとの値の変化を把握するように設計されているため，短時間のジョブについての動きを監視するには適していない．

Visual GXP(VGXP)[7] は，通信量や監視の負荷を抑えつつ 1 秒から数秒間隔で各監視項目の値を集め，各計算機の状態変化を迅速に把握できるように設計された多拠点分散環境向けのモニタリングシステムである．収集した各計算機の情報はそれぞれを比較しやすいように可視化を行い，特別にしきい値を設定しなくとも異常の可能性を視覚的に判断し，必要に応じてさらに詳しい情報を見ることができる．**図 2** にスクリーンショットを示すが，小さい画面に多くの情報を表示できるように，各ノードを 2 次元平面上に配置し，それぞれのリソース利用率を縦棒グラフで表すという，3 次元的な可視化を行っているのが特徴である．収集している情報は， CPU やメモリの使用率， 通信量やディスク I/O の量， アクティブなプロセスの CPU 使用率などで，他の情報を収集するように拡張することもできる．また，クライアントのインストールを簡単にするため， クライアントには Java Web Start を利用している． このため， ブラウザからリンクをクリックするだけでアプリケーションのインストールと起動を行うことができる．

4.3 MC-MPI

MC-MPI (Multi-Cluster MPI)[8] は広域環境用の適応的な MPI ライブラリである．

MC-MPI はファイアウォールや NAT がある環境 (InTrigger の場合 kyoto と imade) でも動作するように，メッセージの中継を行う．このために手動の設定は必要なく，MC-MPI は自動的に通信できるノードを発見してルーティングを行う．

また，MC-MPI は，多数のノードにスケールするように，各プロセスが確立する接続の数を制限する．この際，遠いノードとの接続を削減し，近いノードとは密に接続を張ることによって性能を維持する．ノード間の距離は起動時にピンポンを行うことによって自動的に取得する．

このように MC-MPI は環境に適応するためにルーティングを行ったりピンポンを行ったりするが，高速に起動する．InTrigger の 6 拠点 486 コアでは 12.5 秒で起動した．

MC-MPI を用いると広域環境で UPC (Unified Parallel C) のプログラムを実行することもできる．Berkeley UPC[9] ランタイムは MPI を用いて通信を行うことができるので，MPI ライブラリとして MC-MPI を用いることによって，ファイアウォールや NAT がある広域環境で UPC プログラムが実行できるようになる．

4.4 dds

dds は，分散環境上で容易にプログラムを記述できるように開発された，分散オブジェクト指向ライブラリである．このライブラリでは通常のオブジェクトと遠隔に存在しているオブジェクトに対するアクセスを透過的に扱うことができるため，オブジェクトの位置を気にすることなくプログラムを記述することができる．同期的な RMI はもちろん，非同期的な RMI も呼び出すことが可能である．

従来から RMI を透過的に見せるミドルウェアは数多くあった．しかし，それらは大規模の台数で動作することは想定されておらず，例えば一つのオブジェクトの一つの参照につき，スレッドを一つ，ネットワーク接続を一つ，などという設計になっている．また，並列環境を想定したものであっても，参加している計算資源間で，全対全の接続が張れるという仮定がある．これは Intrigger のような台数の大きいでは大きな制約となり，NAT や Firewall にも対応できない．さらに，計算に使う資源の構成を設定ファイルに記述するなど，アプリケーションを書く人に敷居が高くなっている．

dds では，通信遅延を意識したオーバーレイネット

ワークを構築し，内部でルーティングを行うため，連結であれば通信することが可能である．このため，NATやFirewallが存在する環境下であっても動作し，大規模な台数になってもスケールする．また，事前に計算に参加する資源などを設定ファイルに指定する必要はなく，動的な資源の追加，削除を許す．

オブジェクトの位置が固定されていると，性能上非効率であったり，動的なプロセスの増減に対応できないため，ddsではオブジェクトマイグレーションをサポートしている．この機能はすべてのリモートオブジェクトに対して呼び出すことができ，アクセスの多いプロセスにオブジェクトを移動させることで効率的に実行することができる．

また，現時点では明示的に呼び出さない限り実行されないが，分散GCも実装されている．この分散GCはスナップショットでとったオブジェクトグラフを元にマークスイープを行う分散GCであり，参照されていないすべてのオブジェクトを回収することが可能である．

5. 計算資源利用ルール

InTriggerには様々なジョブを実行する多数のユーザがいるが，現在は資源利用に関するルールなしで各ユーザがジョブを実行している．しかし，このような運用形態では，性能測定のために一時的に計算資源を占有するなどといったことが困難である．一方，バッチスケジューラの利用を義務付けると，小さなジョブも毎回ジョブキューを通さなければならず，煩雑である．

そこで，一時的なら計算資源を占有できるが，ジョブキューの利用は義務付けない資源利用ルールを検討している．この資源利用ルールの基本的な考えは，計算資源を長時間利用しているユーザが，他のユーザがその計算資源を利用しようとしても利用し続けることを「迷惑行為」とみなす，というものである．迷惑行為さえ行わなければどのようにジョブを実行してもかまわないが，様々なジョブを迷惑行為とみなされずに実行するためのツールとして下記のものを提供する．

TORQUE: 短時間計算資源を占有したいユーザのためにバッチスケジューラであるTORQUE[10]を提供する．本ルールではTORQUEのジョブキューを通して実行したプロセスに優先権があるとし，ジョブキューを通さずに実行したプロセスがある程度以上CPUを利用した場合は迷惑行為とみなす．一方，ジョブキューを通して実行したプロセスでも長時間CPUを使い続ける場合は迷惑行為とみなす．また，ジョブキューを通してnノードでプロセスを実行した場合の迷惑行為とみなす時間は，1ノードで実行した場合の時間の$\frac{1}{\sqrt{n}}$倍とする（$\frac{1}{n}$倍とせずに$\frac{1}{\sqrt{n}}$倍とするのは，InTriggerにおいて並列ジョブを推進するためである）．

nicer: 計算がいつ行われてもかまわない代わりに長時間計算を行いたいユーザのためにはnicerを提供する．nicerはniceコマンドの拡張で，計算資源の利用状況に応じてプロセスを一時停止させたり再開させる．TORQUEのジョブキューを通して実行したプロセスが現れたらプロセスを一時停止させ，消えたらプロセスを再開させることによって，迷惑行為を行うことなく空きCPU時間を有効利用する．

デバッグ用の小規模な実行などは，CPUをあまり使わないので，TORQUEもnicerも利用せずに実行しても迷惑行為とみなされることはない．

ユーザにルールを守らせるための仕組みとしては，定期的にpsなどを実行することによって迷惑行為を検出して，ユーザ毎の迷惑行為の週報を作成する．

```
# メッセージヘッダを受信
header = recv(8)
size = get_size(header)

# メッセージ本文を受信
while received < size:
  frag = recv(size - received)
  body += frag
  received += len(frag)
```

図3 バグを含んでいた部分のコード
Fig. 3 The code containing the bug

6. 大規模広域環境におけるソフトウェアの検証

グリッド用のソフトウェアは広域環境で多数のノードにスケールする必要があるが，大規模広域環境におけるソフトウェアの検証は容易ではない．そもそも実験を行うための環境が必要であるが，少数の団体のアドホックなコラボレーションではそこまで大規模で広域な環境を整えることができない．本章では，InTriggerのように多拠点に渡る分散計算機環境を構築することによって初めて行える実験があるということを示し，そのような実験を行うことによって得られた知見について述べる．

例としてMPIライブラリとして広く用いられているMPICH2[11]の検証を行った．MPICH2は特にグリッドを意識したライブラリではないが，ファイアウォールやNATのない環境では動作するはずのものである．バリアを張るだけのプログラムを1クラスタ83ノード(chibaのみ)と3クラスタ83ノード(chiba, hongo, okubo)で実行したところ，1クラスタの場合は毎回問題なく実行できたが，3クラスタの場合は高い確率でデッドロックを起こしてしまった．

デッドロックが生じる直接の原因は，バリアを張るためのsendの中でハングしてしまうプロセスがあるということであったが，実際のバグはそのプロセスとはまったく関係ないところにあった．MPICH2では，各プロセスは他のプロセスと最初に通信を行うときに初めてそのプロセスとの接続を確立する．その際にリング上に接続されているプロセスマネージャを介してエンドポイントを交換するが，デッドロックの元々の原因はこのプロセスマネージャのバグにあった．

プロセスマネージャのバグを含んでいた部分のコードを**図3**に示す．`recv`システムコールは要求したバイト数より少ないバイト数しか受信せずに返ることがあるためループの中で呼ぶ必要があるが，図3のコードではメッセージヘッダを受信する部分にはループがない．ヘッダはわずか8バイトであるため，単一のクラスタという低遅延環境では常にまとめて届いたが，3クラスタを用いた場合はより遅延が高かったため，8バイトでも分かれて届いてしまうことがあった．その結果，ヘッダの一部しか受信せずに本文の受信に進んでしまうことがあった．

このようにInTriggerを用いることによって小規模な環境では生じないバグを見つけることができた．しかし，バグが広遅延環境でしか生じないものであり，また，デッドロックが検出された箇所と実際にバグがあった箇所の間には多数のプロセスが絡んでいたため，流れを手動で追うことは容易ではなかった．これは，このような大規模分散環境における問題追跡を支援する仕組みの重要性を現している．

7. おわりに

本稿では，大規模分散計算機環境InTriggerについて説明した．これまでに6拠点に514コアを導入して，OSの遠隔再インストールが完全に自動的に行えるようにした．また，GXP, VGXP, MC-MPI, ddsなどの分散環境を使いこなすためのツールが利用できるようにした．最終的にInTriggerは20-30拠点1000コアになる予定であり，大規模且つ広域でなければできないような様々な実験ができるようになる．

現在InTriggerでは資源利用に関するルールなしで各ユーザがジョブを実行しているが，今後は提案した迷惑行為に基づく資源利用ルールを運用する予定である．現在のルールではCPUに関してのみ考えるが，最終的にはディスクやネットワークなどのI/Oに関しても考える必要がある．

謝辞

本研究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．本環境を構築するにあたって，NECの高宮氏にLucieの使用方法や修正について支援を頂いた．また，クラスタを設置した各研究室(京都大学角研究室・中島研究室，早稲田大学山名研究室，東京工業大学合田研究室)の皆様にも協力を頂いた．

参考文献

1) Grid'5000: Online at http://www.grid5000.fr/.
2) DAS-3: Online at http://www.cs.vu.nl/das3/.
3) SINET 学術ネットワーク: Online at http://www.sinet.ad.jp/.
4) 高宮安仁, 真鍋篤, 松岡聡: Lucie: 大規模クラスタに適した高速セットアップ・管理ツール, 情報処理学会論文誌：コンピューティングシステム, Vol. 44, No. SIG 11 (ACS 3), pp. 79–88 (2003).
5) FAI - Fully Automatic Installation: Online at http://www.informatik.uni-koeln.de/fai/.
6) Taura, K.: GXP: An Interactive Shell for the Grid Environment, *Proceedings of the 8th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA 2005)*, pp. 59–67 (2005).
7) 鴨志田良和, 金田憲二, 遠藤敏夫, 田浦健次朗, 近山隆: VGXP: 多数の計算機をリアルタイムに監視・操作するソフト, 並列/分散/協調処理に関するサマー・ワークショップ (SWoPP 2006), pp. 19–24 (2006).
8) Saito, H. and Taura, K.: Locality-aware Connection Management and Rank Assignment for Wide-area MPI, *Proceedings of the 7th International IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, pp. 249–256 (2007).
9) Berkeley UPC: Online at http://upc.lbl.gov/.
10) TORQUE Resource Manager: Online at http://www.clusterresources.com/pages/products/torque-resource-manager.php.
11) MPICH2: Online at http://www-unix.mcs.anl.gov/mpi/mpich/.