【パートII：情報分野研究者のためのオンリーワン共有イノベーションプラットフォーム】

4. InTrigger：オープンな情報処理・システム研究プラットフォーム

田浦健次朗[*1]

[*1] 東京大学

概要

InTrigger[☆1]は日本の多くの大学を拠点とした，クラスタ計算機を結合した分散環境である．2005年度末より開発と設置が始まり，2010年度までに，20以上のクラスタを結ぶ予定である．2008年6月現在で10の大学・研究所を拠点とした11のクラスタ環境を構築，運用している（**図-1**, **表-1**）． 計算ノード数は319ノード，CPUコア数は848である．

OS，ディストリビューションは，Debian GNU/Linux[☆2]の最新stable版に統一されており，基本的なソフトウェア構成，カーネルバージョンなども統一されている．利用者がそれらの拠点を一体的に利用することができるように，新パッケージの追加やOSの再インストールなどを，全拠点に対して一斉に行うことが可能な体制で運用されている．

想定利用者層は多岐に渡るが，分散並列処理，ネットワーキングなどのいわゆるシステムソフトウェアの研究者と，自然言語処理，バイオ情報処理，データマイニング，Web解析，センサデータ解析などの，大規模データ処理を必要とする分野の研究者を主な層としている．2008年6月現在で利用者アカウント数は172（異なる研究グループ数50）となっており，それら全利用者が全拠点を利用することが可能である．

InTriggerの理念と目的

InTrigger構想時からの最大の目的の1つは，並列処理を専門としない分野で，むしろその「必要に迫られて」いる研究者が，日々必要な大規模データ処理を当たり前のように行える，生産性の高いソフトウェア，環境，ノウハウを確立することである．そして，同じ環境上にシステムソフトウェア研究者と，その研究成果ソフトウェアの潜在的ユーザを同居させ，それにより開発者—利用者の間で密なフィードバック，連携を可能にすることである．今日のInTriggerでは，Webデータに対する自然言語処理を行っている研究者を中心にして，InTrigger環境で開発された成果を利用しながら，複数拠点にまたがる数百並列程度の処理を日常的に行っており，上記目標はある程度達成されているといってよいであろう．

もう1つの大きな目的は，並列分散処理，ネットワーキングなどシステムソフトウェア研究のための大規模な並列・分散テストベッド環境，およびその成果を利用者へ発信できる環境を構築することである．基本的にすべてのシステムソフトウェアは，どこかにスケーラビリティの限界，ボトルネックを持っており，またおそらくほとんどのソフトウェアはそこで露呈するバグを抱えている．それは数十程度の並列度，1クラスタ内など遅延の短い環境ではまったく問題にならないことも多い[6]．逆に言えば，数百程度の並列度で開発，テストを行うことは今日の並列処理としては必須のものである．また，NAT，firewall，ルータによる接続の自動切断など，広域ネットワーク環境特有の困難もある．InTriggerはロバストで移植性の高い分散ソフトウェアを設計する基盤として，貴重な実環境を提供している．

InTriggerはオープン（開放的）であることを中心的な設計理念としている．その意味は，

オープンなソフトウェア構成・設定：InTrigger環境を構成するソフトウェアにベンダ依存の独自（proprietary）ソフトウェアはなく，すべてがLinuxクラスタ環境において誰でも再構築できるフリーソフトウェアである．これによりInTrigger上での利用プラクティス，ノウハウ，開発努力を今日の標準的クラスタ環境に移行できるものとしている．

オープンな管理体制：InTriggerは複数のグループのボランティア研究者によって管理されており，環境をよ

☆1 http://www.intrigger.jp/

☆2 http://www.debian.org/

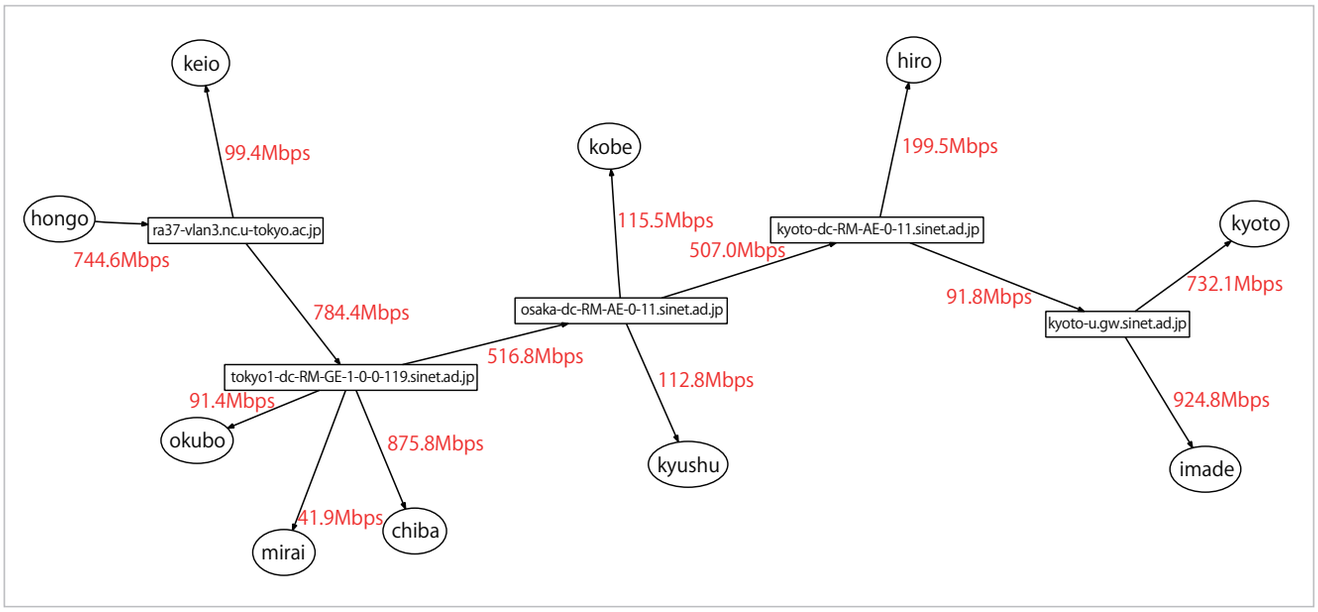

図 -1　InTrigger 拠点（2008 年 6 月）と拠点間トポロジ，バンド幅

通称	導入年度	所在地	ノード／コア数
chiba	2005, 6	NII 西千葉分館	90/147
hongo	2005, 6	東京大学本郷	65/79
imade	2006	京都大学	30/60
kyoto	2006	京都大学	35/70
okubo	2006	早稲田大学	14/28
suzuk	2006	東工大すずかけ台	36/72
mirai	2007	はこだて未来大	6/48
kobe	2007	神戸大	11/88
kyushu	2007	九州大	10/80
hiro	2007	広島大	11/88
keio	2007	慶應義塾	11/88
計			319/848

表 -1　InTrigger 拠点（2008 年 6 月）

くするために管理者グループへ積極的に参加する利用者を常時歓迎している．そして，管理者は各拠点ごとにその拠点のクラスタだけを管理するのではなく，遠隔のクラスタを，OS の再インストールを含めて管理・設定することができるという方針をとり，かつ実際にそれが可能な体制を整えている．そのため研究上の必要に柔軟・迅速に対処することが可能であり，納入業者の都合で必要なソフトウェアが使えない，などということはない．

オープンなネットワーク：多数の拠点をまたぐ分散プラットフォームとして設計された InTrigger では，各拠点の協力を得て，拠点をまたぐノード間通信をほとんど制限することなく行うことができる．InTrigger の外部との通信制限も，拠点の協力を得て，ほとんどの拠点においては拠点出入り口のルータではなく，ホストごとの設定（iptables）で行われている．これにより，各拠点のネットワーク管理者の手を煩わせずに，新しい拠点の追加や，研究上の必要に応じた新拠点との接続（フィルタリング解除）を，迅速に行うことができる．

総じて InTrigger は，「多くの実験では，1 クラスタ内にすべての資源が閉じることはなく，クラスタにまたがる処理が必要であること」「利用者は，たとえ強力な環境であっても，1 つの環境だけで通用するソフトウェアやノウハウを求めていないこと」を基本認識として設計されている．たとえば大規模データ処理研究では，処理すべきデータがその利用者の拠点にあり，事前にすべてのデータを計算拠点内に動かしておくことは現実的ではない，処理すべきデータは WWW から on-demand に取り出される，計算の一部で大量のメモリを消費するのでそのための専用ノードと連携する必要がある，などの状況が頻繁に生ずる．InTrigger とその上のシステムソフトウェアはそのような状況で，柔軟に他の拠点と連携できる，複数拠点にまたがった計算を容易に実行できることを目指して設計されている．

利用者環境

利用者環境の設計にあたっては，

- 前章で述べたような開放性を持っており，拠点間をまたがる分散並列処理，そのためのプログラム開発を柔軟に，効率よく行える環境であること，
- 不安定なソフトウェアに依存せず，安定して運用でき

ること
- システム全体で単一故障点や単一故障拠点を持たず，拠点ごとに独立して稼働できること

を最重要基準としている．

利用者には全拠点同一のアカウント名が提供され，SSH を用いてログインが行えるという，見慣れた Linux クラスタ環境が提供される．アカウント共有は拠点ごとに NIS を用い，ファイルシステムは拠点ごとに NFS を用い，ホームディレクトリと一部のデータディレクトリ（拠点につき 10TB 程度）を拠点内で共有する．そのほかにデータ領域として各ノードのローカルディスクが 1 ノードにつき 500GB から 2TB 程度提供されている．

拠点にまたがったファイルの共有に関しては，特定の分散ファイルシステムに依存した運用はしておらず，個々の利用者が任意に利用，または開発できる環境を提供している．まず，ユーザレベルでファイルシステムを構築することができる枠組みとして UNIX で標準的になりつつある FUSE（Filesystem in Userspace）☆3 と，それを用いて SSH ログインが可能な任意のノードの任意のディレクトリをローカルファイルシステムにマウントできる，SSH Filesystem ☆4 が全ノードにインストールされている．これにより InTrigger から利用者拠点にあるデータにアクセスする，またはその逆などを容易に行うことができる．また，分散ファイルシステム Gfarm ver.2 [2] ☆5 が導入，運用されており，InTrigger の，ほぼ全ノードのデータ領域を拠点間で共有することができる．Gfarm もこれを実現するために一部で FUSE を用いている．Apache webserver が全ノードにインストールされており，計算結果に Web ブラウザでアクセスしたり，wget などのコマンドでファイル転送を Web サーバ経由で行うことができる．

大規模な並列環境や分散環境でのファイル共有には，性能，安定性，NAT や Firewall 対策，管理のしやすさ，タスクスケジューリングとの連携など，研究の余地がある．特に安定性についてはまだまだ改善の必要があるだろう．InTrigger は，現時点で管理者がどれかにコミットするというよりも，それらの研究に必要なパッケージを導入し，研究が発展するための実験環境を提供しようとしている．

☆3 http://fuse.sourceforge.net/

☆4 http://fuse.sourceforge.net/sshfs.html

☆5 http://sourceforge.net/projects/gfarm/

☆6 http://www.clusterresources.com/pages/products/torque-resource-manager.php

☆7 http://www.logos.t.u-tokyo.ac.jp/gxp/

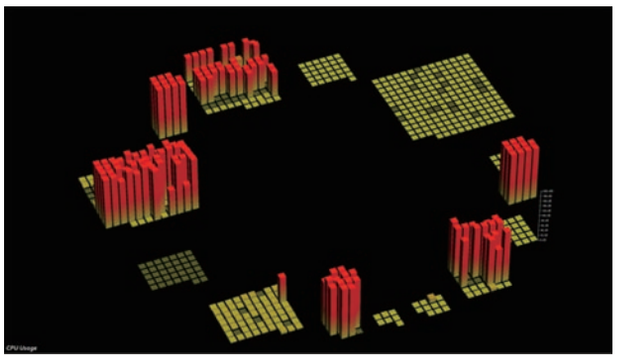

図 -2　Visual GXP スナップショット

ジョブ実行環境としては，拠点ごとにバッチスケジューラ TORQUE ☆6 が導入されており，qsub コマンドなどでバッチジョブ（スクリプト）を投入することができるが，SSH を用いて他のノードへログインし，個々のノードでプロセスを立ち上げることも可能である．並列処理のためのツールとしては筆者による GXP [3] ☆7 が多くの利用者に用いられている．

利用者が，InTrigger 環境全体の状況を容易に把握できるようにするために，GXP 上のモニタリングツール Visual GXP [1] が日常的に使われている．**図 -2** は Visual GXP のスナップショットであり，Java Web Start の仕組みを使って，Web ブラウザから容易に表示可能である．個々のバーが 1 ノードの CPU 利用率を表しており，メモリの使用量，ネットワーク使用量，個々のノード上のプロセスなどを表示することも可能である．それらが実時間（数秒の遅れ）で表示されるため，システム全体の混雑度の把握，空いているクラスタの選択，暴走プロセスの発見，並列処理の負荷分散の把握などに，きわめて有効に使われている．このほか，ノードの稼働状態の把握のためにモニタリングシステム ganglia，munin，自動障害通知のために nagios を運用している．

プログラム開発に関連するパッケージとして，erlang，gcc，gfortran，g++，jdk，ocaml，perl，python，ruby，tcl などのプログラミング言語，sqlite，postgresql，mysql などのポピュラーなデータベースを揃えている．

InTrigger の構築と運用

InTrigger 環境の構築にあたっては，

1．すべての拠点の OS，インストールパッケージなど

を統一すること，
2. 必要に応じて全ノードへのパッケージの追加，カーネルの更新などが迅速に行えること，

を 2 大要件としている．要件 1 が満たされていない環境は，あるタスクが一部のノードでのみ失敗するといった生産性低下の原因になる．特にバッチキューイングシステムなどの非対話的な環境で自動的にスケジュールされたジョブの場合に，この生産性の低下は著しい．そしてそのような状態は実際，一部のノードが故障しているときに構成変更を行ったときなど，日常的に生ずる．要件 2 によってある程度頻繁にパッケージが更新されると，故障で一時的に外れたノードを含めてパッケージの状態を同じに保つのは ad-hoc な方法では困難で，何らかのシステム化された構成情報管理が必須になる．

InTrigger ではこれらの目標達成のため，

- 全拠点の遠隔管理
- 全拠点の構成情報の一元管理
- 全拠点の OS 一斉インストール，構成情報の適用

を行っている．構成情報は単なるパッケージ情報だけでなく，ハードディスクのパーティショニング，ネットワークインタフェースの設定など，OS インストールに必要な情報がすべて含まれている．一部のノードの OS が起動不可能，またはネットワークの構成が失敗してネットワークから到達不可能な状態になっても，電源をリセットしたり，起動時のコンソール画面を見られたりするように，Intelligent Platform Management Interface（IPMI）を用いている．

OS の一斉インストールには，高宮による Lucie[5] ☆8 を用いている．Lucie サーバが OS のインストーラを NFS によってノードに提供するため，CD-ROM などの媒体をセットすることなく遠隔から OS がインストールできる．ネットワークからの起動（PXE）が可能なマシンであれば Lucie を用いたインストールが可能である．1 ノードの再インストールにかかる時間は数分であり，台数が増えてもほぼ一定である．

ソフトウェアの構成管理には Puppet ☆9 を用いている． InTrigger の設定情報は，ハードディスクのパーティショニングや必要なモジュールのコンパイルや起動，ネットワークインタフェース，IP フィルタリングの設定等，低レベルな設定情報から，インストールするパッケージやその設定ファイルなどに至るまで Puppet を使って記述され，一元管理されている．これらにより InTrigger ノードは，OS のインストールからソフトウェアの構成管理までを，全拠点の全ノードに対して，遠隔から行うことが可能である．

☆8 http://lucie.is.titech.ac.jp/trac/lucie/

☆9 http://reductivelabs.com/trac/puppet/wiki

GXP

GXP は，分散環境の管理，日常利用から並列処理までを快適に行うために，筆者が 2003 年度から開発を始めたツールで，2 度のスクラップ & 再実装を経て現在に至っている．Python 処理系さえあれば 1 ノードに導入するだけで，複数クラスタを用いた並列処理が可能になる．導入にはルート権限も必要なく，コンパイルする必要もない．

初期の目標は，分散システム用のいわゆる並列シェルで，多数のプロセスを一斉に，高速に立ち上げることであった．遅延の大きい分散環境であっても，数百プロセスの立ち上げから出力，終了までを数秒で行うことができるため，大規模な，いわゆる「重い」並列処理のみならず，負荷平均やメモリ使用量を調べるなどの診断からシャットダウンや設定ファイルの配布などの管理業務に有用であった．基本的な仕組みは SSH のみを基本にしており，高速性とスケーラビリティのために以下がポイントであった．

1. SSH で一度ログインを行ったホストに対して接続を維持しつづけ，その後のコマンド実行時にはその接続を通してコマンドを高速に実行すること，
2. 多数のホスト間の接続を，ローカルホストを根とした木構造で維持したこと．これにより，数百ノードと接続を維持しても，ルートノードに生成される SSH プロセスは高々数個となり，負荷が少ない．

現在はそれらの性質を維持したまま，以下のように機能や目的を拡張している．

1. SSH の代わりに TORQUE や SGE などのバッチスケジューラを通してホスト間の接続を作ることができ，バッチキューイング環境でも使うことができる．SSH や異なるバッチキューイングシステムが混在していても使うことができる．
2. 結果として，特段の困難なく（SSH と，場合によりバッチキューイングシステムを組み合わせて）複数拠点を組み合わせて使うこともできる．
3. Workflow 実行の仕組みとして，GXP と，無変更の GNU make を組み合わせて並列 make を分散環

境で実行することができる．

これにより，多様な環境を統一的操作インタフェースに用いることができる環境，バッチキューイングシステムを対話的レスポンスで用いることができる環境，分散環境で手軽に並列 workflow を実行できる環境，として強化されている．

総じて，GXP は管理方式やジョブ投入の方法がまちまちである複数の環境を統合し，統一的な操作インタフェースを提供するプロセス管理，スケジューリングレイヤとみなすことができるようになっている．これは，複数の拠点を，拠点スケジューラ間の直接的な連携なしに用いる手法として将来性のあるモデルである．たとえば，文献 4）では，バッチスケジューラである Condor システムを複数拠点にスケールさせる際に，複数拠点のスケジューラを直接連携させるのではなく，Condor のプロセスマネージャ自身を，他の拠点へ通常のジョブとして投入してしまう，gliding in，もしくはオーバレイスケジューラという考え方が有効であったことが報告されている．GXP で複数拠点を同時に使った並列処理を行う仕組みもこれと同様であり，まずバッチスケジューラを用いて必要なノードとの接続を作っておき，それらを利用可能な計算資源として個々のプロセスを分散させる．

関連研究

並列処理環境は，大別して純粋な利用者に対する計算資源を提供する環境（大学の計算機センターなど）と，システムソフトウェアやネットワーキング研究のためのテストベッド環境とに大別される．

前者は資源の厳格な管理と安全性を最優先して設計されているのが通常で，計算ノードから拠点外アクセスできないなど，複数拠点間の通信も非常に制限されていることが多く，本稿中で述べたような新しいシステムソフトウェアの利用や研究に適した環境とは言えない．したがって利用者層も計算機科学，情報科学分野というよりは，数値シミュレーションなどの計算科学・工学分野に限定されてきた．

後者の実例としては，ネットワーキング研究を主目的とした PlanetLab[☆10]，StarBED[☆11] などがある．PlanetLab は世界中の数百拠点を結ぶテストベッドで，広域ネットワークの実環境として実績がある．ただし拠点ごとのノードは 1 ～数ノード程度で，基本的に 1 研究グループが 1 ～数ノードを自主的に拠出するという運用形態をとっている[☆12]．広域ネットワークのテストベッドとしては優れているものの，大量の資源（CPU，ネットワーク，メモリ，ディスク）を用いた計算を行うことは困難で，また想定もされていない．StarBED はネットワーク研究のための実験環境で 600 以上のノードからなる．資源は 1 拠点に集約されておりネットワークを仮想的にエミュレートする機構により，さまざまなネットワーク研究に適した環境を構築することを可能にしている．どちらも計算資源としての利用が主目的である利用者は対象にしておらず，仮想化された計算機を元に実験環境の構築からすべてを行う利用者を想定している．

計算機科学研究を目的とし，したがって目的や構成は InTrigger と最も近いプラットフォームに Grid'5000[☆13]，DAS-3[☆14] などがある．Grid'5000 はフランスの 9 拠点からなる Grid 研究のためのプラットフォームである．各ユーザが自分用の環境を構築・維持するという方式で運用されており，実験にあたっての柔軟性は大きいものの，実験を始める人的コストが大きく，計算資源としての利用が主目的である利用者にとっての利便性は低い．DAS-3 は，利用者が共有して利用する環境を常時運用しており，利用者から見た環境としては InTrigger に最も近い．用途はシステムソフトウェア研究に焦点を当てており，DAS-3 の Web サイトによれば，通常は 1 ジョブを 15 分程度で終了させることを基本としているようである．

まとめと今後の展望

並列処理は長い研究の歴史にもかかわらず利用者層は限られてきた．しかし最近になって，マルチコアプロセッサ，AmazonEC2 のようなクラウドコンピューティング環境など，利用者が「並列処理を行う」利益が身近になりつつある．米国では，IBM と Google が提携して，大学の計算機科学関係の専攻に対して大規模クラスタを提供する研究支援の取り組みを始めており，計算機科学に携わる者に並列処理をますます身近なものにしようとしている．日本の高性能並列計算機も，東工大 TSUBAME，筑波大，東大，京大の T2K など，上位はことごとく Linux クラスタとなり，「通常の利用者環境」と，高性能計算機の間の不必要な非互換性はますます少なくなっている．

☆10 http://www.planet-lab.org/

☆11 http://www.starbed.org/

☆12 執筆時点で 461 拠点，893 ノード．

☆13 http://www.grid5000.fr

☆14 http://www.cs.vu.nl/das3/

そのような状況で今後，並列処理に対して一般利用者から見て重要な評価基準，つまり，性能一辺倒ではない生産性，移植性，信頼性に対する要求が高まってくるのは必然の流れであろう．また，用途に応じてworkflowやスクリプト言語など，高水準での並列処理が行えることも重要性を増すと考えられる．MapReduceのような問題領域に特価したプログラミングモデルや，Google App Engineのような目的を限定した，だがシステム構築や設定の手間がかからない環境が注目を集めていることもその表れといえる．

今後の計算環境は，広範囲の利用者層に対して，望む環境を即座に構築し，提供できるような環境へと進化していく必要がある．InTrigger環境を通じて我々が行ってきた研究や，参加研究者の成果でInTrigger上に展開されているものには，環境を構築する汎用的なツール（Lucie），分散環境を容易に結合して快適な並列処理を可能にする汎用的なツール（GXP），並列スクリプト言語（gluepy☆15），分散ファイルシステム（Gfarm）などがあり，いずれも，この進化の方向性に適合した努力であると考えている．今後もInTriggerは，情報爆発時代の並列処理に対して，環境構築，利用モデル，プログラミング，その処理系，分散データ共有など多方面にわたる新しいアイディアを育み，実現する環境としての役割を，今後も果たしていきたいと考えている．

☆15 http://www.logos.ic.i.u-tokyo.ac.jp/~kenny/gluepy/

謝辞 InTriggerは拠点を提供している研究者の皆様，ならびに日頃InTrigger環境の運営に尽力している研究者（主に学生，ならびにInTrigger環境の構築に深くコミットしていただいたLucieの作者NEC 高宮氏）の協力なしには決して動かない．この場を借りてお礼を申し上げたい．

参考文献

1) Kamoshida, Y. and Taura, K. : Scalable Data Gathering for Real-time Monitoring Systems on Distributedcomputing. In *8th IEEE International Symposium on Cluster Computing and the Grid* (2008).
2) Tatebe, O., Soda, N., Morita, Y., Matsuoka, S. and Sekiguchi, S. : Gfarm v2 : A Gridfile System that Supports High-performance Distributed and Parallel Data Computing. In *Proceedings of the 2004 Computing in High Energy and Nuclear Physics* (*CHEP04*)(2004).
3) Taura, K. : GXP : An Interactive Shell for the Grid Environment. In *Innovative Architecture for Future Generation High-Performance Processors and Systems*, Vol.12-14, pp.59-67 (2004).
4) Thain, D., Tannenbaum, T. and Livny, M. : Distributed Computing in Practice : The Condor Experience. *Concurrency and Computation: Practice and Experience*, 17 (2-4) : pp.323-356 (Feb.-Apr. 2005).
5) 高宮安仁，弘中 健，斉藤秀雄，田浦健次朗：複数拠点に分散配置されたクラスタの効率的な管理手法．In *コンピュータシステム・シンポジウム* (*ComSys 2008*) (2008).
6) 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：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．In *情報処理学会研究報告 HPC-111*(*SWoPP 2007*)，pp.237-242 (2007).

（平成20年7月9日受付）

田浦健次朗（正会員）
パートII「0. 前書き～研究者が作る研究基盤」を参照