グリッドコンピューティングのための小型計算機システムの開発

代表研究者　義久智樹　京都大学 学術情報メディアセンター 助教
共同研究者　金澤正憲　京都大学 学術情報メディアセンター 教授
〃　岩下武史　京都大学 学術情報メディアセンター 准教授

1 はじめに

近年，ネットワークを介して接続された複数の計算機を用いて処理を行うグリッドコンピューティングに対する注目が高まっている．複数の計算機で並列処理することで，個々の計算機で処理するよりも大容量の情報を高速に処理できる．ほとんどのグリッドコンピューティングシステムでは，パーソナルコンピュータやワークステーションといった,ある程度の処理能力をもつ計算機を用いており,GLOBUS[1] や UNICORE[2], NAREGI[3]といったグリッドコンピューティングのためのミドルウェアが開発されている.これらのミドルウェアは,小型計算機を対象としていないが,多数の機器に組み込まれている小型計算機と連携させることで，計算資源や搭載されているセンサを有効利用できる．小型計算機と大型計算機を連携させて行う処理として次のようなものが考えられる．

・ケース 1

IC レコーダーのようにマイクから音声データを入力し保存可能なものがある．これに大型計算機を利用することで逐次的な文字データへの変換や，ノイズ除去などの処理能力を必要とする処理を複合的に行うことができる．

・ケース 2

小型計算機にはセンサノードや，各種機器に搭載されている温度センサを有するものがある．様々な場所の小型計算機を利用して大型計算機に温度センサの値を収集し，大規模または綿密な環境のモニタリングを行う．

これまで，大型計算機や中型計算機を用いたグリッドコンピューティングは，データ送信手法に関する研究[4]，超高圧電子顕微鏡を接続した研究[5]といったように研究用途で利用されているものの，専門の知識が必要になり，研究者以外がグリッドコンピューティングを利用することは困難であった．また，組み込み型の小型計算機はその機器の機能を実現するのが主目的である．近年，インターネット家電のようにインターネットに接続する機器が登場してきているものの，これらにおいては大型計算機と小型計算機の処理の連携は行われていない．しかしながら，ケース 1 やケース 2 のように大型計算機と小型計算機が処理を連携することは有効である．そこで，本研究では，日常生活における多数の機器に組み込まれている小型計算機を用いたグリッドコンピューティングを提案し，実現のためのシステムを開発する．

2 グリッドコンピューティングのための小型計算機システム

一般に，機器に組み込まれた小型計算機は，普段は割り当てられた通常の処理を行っており，他の処理を依頼された場合でもできる限り通常の処理を妨げることなくその処理を行う必要があった．小型計算機で実行される処理として，あらかじめ割り当てられた通常の処理を通常処理，グリッドコンピューティングシステムから依頼されたその他の処理を拡張処理として規定する．小型計算機を用いたグリッドコンピューティングでは，普段は通常処理を行っており，処理能力が低く，記憶容量が少ないといった特徴から，これまでのグリッドコンピューティングとは異なった枠組みが要求される．以下に小型計算機を用いたグリッドコンピューティングの要件を列挙する．

・処理能力の制御

通常処理を妨げないように，小型計算機の処理能力を制御できる必要がある．グリッドコンピューティングシステムは，小型計算機の処理能力のうちどの程度を通常処理や拡張処理に割り当てるか決定する．

・他計算機との連携

連携のため，小型計算機の識別子やハードウェア構成を記述したプロファイルが必要になる．

・簡単な通信プロトコル

小型計算機は，一般に，処理能力が低く，記憶容量が少ないため，簡単な通信プロトコルを用いてデータの送受信を行う必要がある．

・ 拡張処理の記述

拡張処理は，小型計算機がグリッドコンピューティングシステムから依頼されて実行するため，記述できる必要がある．処理能力や記憶容量の異なるさまざまな小型計算機でも拡張処理が適切に実行されるような記述言語を設計が必要となる．

本研究では，前述の要件を満たしたコンパクトグリッドシステムの提案を行い，その設計と実装を行った[6]．「処理能力の制御」については，通常処理および拡張処理への処理能力の割り当てを動的に変更できるマイクロコンピュータのようなプログラミング可能な小型計算機を用いる．コンパクトグリッドを構成する小型計算機をコンパクトグリッドクライアントと呼ぶ．「他計算機との連携」，「簡単な通信プロトコル」については，これらの要件を満たすコンパクトグリッドミドルウェアと呼ぶミドルウェアを開発した．コンパクトグリッドミドルウェアの次のように構成されている．通常処理と拡張処理は「処理部」で処理され，通常処理は「通常処理制御部」，拡張処理は「拡張処理制御部」で制御される．また，処理部の処理能力の割り当てを行う「処理能力制御部」，コンパクトグリッドアセンブラの受信や他のコンパクトグリッドクライアントとの通信を担当する「通信制御部」から成る．「拡張処理の記述」については，コンパクトグリッドクライアントの処理能力を考慮し，アセンブラ程度の低級な処理記述言語としてコンパクトグリッドアセンブラを開発した．様々なハードウェア構成をもつコンパクトグリッドクライアントに柔軟に対応できるように，アセンブラで拡張処理は記述される．コンパクトグリッドシステムの構成図を図 1 に示す．小型計算機であるコンパクトグリッドクライアントは，マイクロコンピュータや組み込み型計算機を想定している．TinyOS[7] のような組み込み型 OS があらかじめ搭載された小型計算機もある．コンパクトグリッドミドルウェアは，コンパクトグリッドクライアント上で動作し，コンパクトグリッドアセンブラを解釈する．また，埋め込み型の小型計算機を想定したコンパクトグリッドクライアントも試作した．試作したコンパクトグリッドクライアントのプロトタイプを図 2 に示す．小型計算機として Microchip Technology 社のマイクロコンピュータ PIC16F877 または PIC18F452 を用いている．またインターネットを介して制御するために LAN コントローラ，状態表示用の LED や LCD を備える．コンパクトグリッドミドルウェアは Iosoft 社の小型 Web サーバ[8]をもとに実装を行った．インターネットを介して，制御することを考え，通信には HTTP プロトコルを用いている．また，試作したコンパクトグリッドクライアントを用いて，複数のコンパクトグリッドクライアントを用いた処理や大型計算機との接続の確認をコンパクトグリッドシステムを利用するためのソフトウェア例を示して行った[6]．

3 処理記述言語の設計

コンパクトグリッドシステムではコンパクトグリッドクライアントと中型計算機や大型計算機との連携を考えている．コンパクトグリッドクライアントは計算能力とセンサ類を接続するための入出力端子を備えており，中型計算機，大型計算機からコンパクトグリッドクライアントのリソースを利用する形態を想定している．コンパクトグリッドシステムを用いることによって大型計算機と小型計算機をネットワークで接続することができる．このシステムに対してジョブを投入するには現状ではそれぞれ異なる方式で処理の依頼を行っている．例えば，コンパクトグリッドクライアントである小型計算機は HTTP プロトコルを使用して拡

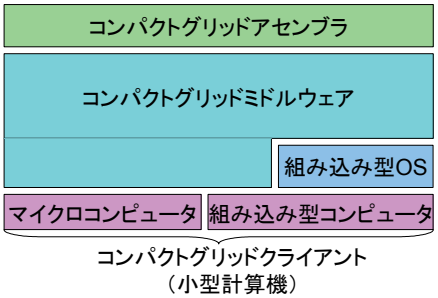

図 1：コンパクトグリッドシステム

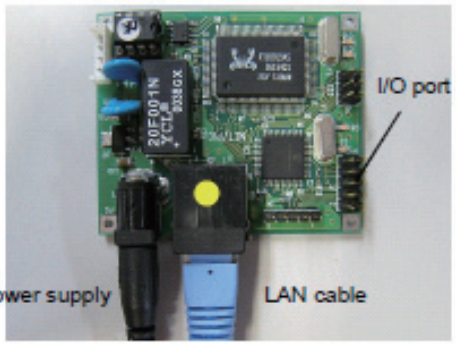

図 2：コンパクトグリッドクライアントのプロトタイプ

張命令の実行を行うようになっている．一方，大型計算機は SSH(Secure SHell) プロトコルでアクセスし，大型計算機に保存してある実行ファイルを実行することで処理を行う．このため，処理を連携させるには，それぞれ異なる方式でジョブの投入・命令実行を行う必要があった．しかしながら，これでは汎用性に欠け，処理連携システムの柔軟かつ広範な運用を阻害する要因となっていた．そこで，小型計算機と大型計算機の処理の連携を用意に利用できるシステム基盤として簡便な処理記述言語を用いることで解決を図る．以下に，処理記述言語の要件を列挙する．

・簡潔な構造

　小型計算機を用いるため，連携処理記述言語は小型計算機でも処理できる程度に簡潔である必要がある．

・処理連携が容易

　大型計算機と小型計算機を連携させるために煩雑な設定の記述を要求せず，必要な項目のみの簡潔な記述である必要がある．

・小型計算機用命令の使用が容易

　小型計算機の命令にはハードウェアリソースを操作できるものもあり，それを利用できる必要がある．

・柔軟な処理命令の記述

　これまで使用してきた言語や言語規則に近いまたは同等の言語を利用できる必要がある．

これらの要件を満たすための言語を提案した．紙面の都合上，代表的なものを図 3 に示す．設定部，大型計算機処理部，小型計算機処理部，出力部の大きく 4 つから構成し，簡潔な構造とする．設定部で大型計算機と小型計算機に関する必要十分な設定を記述するだけで，処理連携システムを構築し，各処理部に記述された内容が実行される仕組みとなっているので，連携が容易という要件を満たしている．また，小型計算機処理部に記述される処理命令は小型計算機の持つ機械語と 1 対 1 で対応するようになっており，小型計算機の命令を簡単に使用できる言語としてコンパクトグリッドシステム提案時に実装を行ったコンパクトグリッドクライアントを用いることを考える．コンパクトグリッドアセンブラの複数の命令を組み合わせることによって，より複雑な動作が実現できる．大型計算機処理部にはシェルスクリプトを用いることができるようにする．なお，連携処理で主要な演算を担う大型計算機での処理は大型計算機であらかじめ実行ファイルを生成・格納しておき，この実行ファイルを実行することで行う．こうすることで，シェルスクリプトでは実現が困難な大規模で複雑な処理を行わせることができ，大型計算機の持つ並列処理などの機能を有効に利用できる．また，主要な演算部分のプログラムは大型計算機が対応しているプログラミング言語を使って書くことができ，既存の実行ファイルも用いることができるので，処理連携システムの利用者の負担を軽減することができる．提案言語は手続き型の言語ではあるが各処理部においては，逐次処理の記述が可能なように設計しており，容易かつ柔軟性の高い処理命令の記述を実現している．処理記述言語の主な命令を表 1 に示す．大型計算機や小型計算機のハードウェア構成は，個々に異なっている．このため，処理に応じて適切な大型計算機やコンパクトグリッドクライアントを指定する必要がある．本研究では，スクリプトの作成者が処理連携システムの構成を把握した上で記述すると考える．

設定部

命令	意味
settings	設定部ブロックの開始
mode	file ファイルモード：ジョブ投入マシンで指定されたデータ量をバッファリングし，データを送信 stream ストリームモード：通信セッションを確立・維持，連続的なデータ送信
target	コンパクトグリッドクライアントのアドレス (IP アドレス or ホスト名)
registry	コンパクトグリッドクライアントのレジストリ番号 (1〜16)
interval	コンパクトグリッドからのデータ取得周期 (msec)
send	データの処理を行う大型計算機のアドレス (IP アドレス or ホスト名)
amount	(ファイルモード時のみ) 1 回の送信データ数
end	設定部ブロックの終了

小型計算機処理部

命令	意味		
process for CGC	小型計算機処理部ブロックの開始		
処理文	コンパクトグリッドアセンブラを使用 (以下は一部)		
CLRW	： ワーキングレジスタのデータを 0 にする．	CNGP a	： 処理能力の割り当てを通常処理 a%，拡張処理 $100-a$%とする．
MOVLW a	： データ a をワーキングレジスタに格納する．	GOTO a	： a 行目にジャンプする．
ADDRR a, b, d_{est}	： レジスタ a と b の和を d_{est} に格納する．		
ADDWR a, d_{est}	： ワーキングレジスタとレジスタ a の和を d_{est} に格納する．		
end	小型計算機処理部ブロックの終了		

大型計算機処理部

命令	意味
process for HPC	大型計算機処理部ブロックの開始
処理文	利用する大型計算機で使用可能なシェルスクリプトで記述
end	大型計算機処理部ブロックの終了

出力部

命令	意味
output	出力部ブロックの開始
Address	出力先のアドレス or ホスト名
Address;RegNum	出力先のコンパクトグリッドクライアントのアドレス or ホスト名，";" 以下に結果を格納するレジスタ番号を指定
end	出力部ブロックの終了

図 3：主な処理記述言語

図 4：処理連携システムの構成

4 コンパクトグリッドシステムの実装

4-1 ハードウェア

処理連携システムの一例として構築を行った(図 4)．大型計算機として京都大学学術情報メディアセンターのスーパーコンピュータ HPC2500 を利用する．HPC2500 は，12 台の計算ノード(フロントエンドプロセッサ 1 ノード+ バックエンドプロセッサ 11 ノード) および外部ディスクへの I/O 処理を行うための I/O ノードから構成される SMP(Symmetric MultiProcessor) クラスタシステムである．ノード間は高速光インターコネクトを用いて高速接続されおり， I/O ノードにはディスク装置が接続されている．中型計算機は処理記述言語を実行し，処理連携システムに対してジョブを投入する端末となる．この部分は一般的なパーソナルコンピュータを想定している．中型計算機として，ノート型のパーソナルコンピュータ IBM ThinkPad X41 を利用した． 各コンパクトグリッドクライアントは中型計算機(構築したシステムではノート型パーソナルコンピュータを使用した)に HTTP プロトコルを利用して，接続されている．中型計算機と大型計算機は SSH プロトコルを利用して接続を行う．

4-2 ソフトウェア

各部をブロックと考えて手続き型言語として実装を行った．処理連携システムを構築を記述する設定部と結果の出力を設定する出力部は非逐次処理とし，必要な設定項目の値を記述するスタイルを採用した．一方で，大型計算機処理部と小型計算機処理部はインタプリタ型を採用している．インタプリタ方式の特徴として，ソースコードを逐次解釈して処理を行うため，プログラムの作成者は実行の順序（シーケンシャルな動作）を意識して記述しやすいことが挙げられる．その他，インタプリタ方式を採用することの利点として，プログラムを作成・変更してから実行ファイルを生成する手間を省くことができる点などが挙げられる．データの送信モードとして，ファイルモードの実装を行ったこれは，プロトコルの違いなどにより，容易にアクセスができない計算機同士の処理連携も考慮したものである．今回，構築したステムの大型計算機も，大型計算機上にあるデータもしくは生成するデータを対象にして大規模高速演算を行う仕様となっている．そこで，大型計算機へのデータの送信はデータを一度ファイルとして出力するファイルモードを使用することで可能となる．なお，ファイルの送信は SCP コマンド(Secure CoPy) を使って送信するという形態をとった．小型計算機処理部ではコンパクトグリッドアセンブラを使用するように実装を行った．コンパクトグリッドアセンブラはコンパクトグリッドシステムにおいて拡張処理を記述するための言語として実装済みである．このアセンブラは機械語と 1 対 1 で対応しており，小型計算機の処理すなわち拡張処理を記述する言語としては最適である．大型計算機処理部ではシェルスクリプトで記述を行い，SSH プロトコルを使用したリモートコマンドの実行を行うようにした．処理をコマンドラインから実行するのと同様のことが行えるようになっており，大型計算機の機能を存分に使用することが可能である．出力部は出力先を指定する．小型計算機の場合は“&” で区切り，レジスタ番号を指示することで，小型計算機の指定されたレジスタに結果の格納を行う．

5 応用例

プロトタイプには，幾つかの入出力端子がある．これらの端子に他の機器を接続し，通常処理による制御を行う．例えば，図 5 に示すように，扇風機をコンパクトグリッドクライアント 1 (CGC1) に接続する．温度センサを CGC1 に接続し，通常処理では，暑いときに扇風機を勢いよく回転させ，涼しいときにはゆっく

図 5：プロトタイプの使用例

りと回転させる．拡張処理では，例えば，30 個のデータの総和を計算する．本研究では，システムの動作確認を目的とし，非常に簡単な計算である総和計算で実験を行う．プロトタイプのレジスタは 16 個であるため，総和計算を CGC1 と CGC2 で分担して行う．パーソナルコンピュータは CGC1，CGC2 とネットワークハブを介して接続されており，データの送信，拡張処理の依頼を行う．実験の結果，正常に総和を計算されていることを確認した（ただし，各 CGC 担当分の総和）．拡張処理を行うことで，通常処理の処理能力が低くなるため,温度に対する扇風機の回転のレスポンスが遅くなることが考えられるが,計算時間が非常に短く，目視による確認はできなかった．

5-1 分散処理

多数の機器に組み込まれている小型計算機を使って処理を分散できる．例えば，第 1 章のケース 1 のように，従来は中型の計算機で行っていた処理を，小型計算機の処理能力を有効利用して行える．しかし，小型計算機の処理能力は貧弱なため，処理の高速化は大きく望めないと考える．小型計算機のような処理能力が低く，記憶容量が少なくても行えるような，少量のデータに対する繰り返し演算を並列に処理できる場合には有効であるといえる．大容量の複雑な演算の場合，大型計算機が適しており，処理に応じて依頼先の計算機を選択することで，計算資源の有効利用が可能になる．

5-1 データ取得

小型計算機に接続されている機器からデータを取得して処理に利用できる．例えば，第 1 章のケース 2 のように，通常処理で用いられているセンサを拡張処理で利用する．データ取得のためのセンサという観点からは，中型・大型計算機のハードウェア構成はほぼ同じであるため，さまざまなデータが必要な場合には，小型計算機を用いたコンパクトグリッドシステムが有効であると考える．小型計算機でデータの取得および補間などの簡単な計算を行い，大型計算機でシミュレーションを行うといった使い方が考えられる．

6 関連研究

関連する研究は，小型計算機に関する研究とグリッドコンピューティングに関する研究に大別できる．

小型計算機に関する研究は，ユビキタスコンピューティング環境の注目の高まりにともない，近年盛んに行われている[9]．Ahrod では，ECA ルールと呼ばれる簡単な記述言語で小型入出力装置の動作を記述し，環境情報の取得や通信を行っている[10,11]．ECA ルールは，動作する事象（イベント），動作の条件（コンディション），動作（アクション）を 1 つの組にして記述した簡単な言語であり，複数のルールを組み合わせることで複雑な動作を実現している．データ容量が小さいため，小型計算機の動作記述に適している．U3 では，幾つかのセンサが取り付けられた直方体の装置を用いて温度や照度を取得する[12]．センサの制御や通信は小型計算機で行われている．小型計算機を用いている点が本研究と類似しているが，U3 ではグリッドコンピューティングを考慮して設計されていない．Smart-Its[13]や Motes[14]といった小型のセンシング装置が開発されているが，いずれも温度や照度といった環境情報を取得するための装置であり，計算処理を依頼して並列処理するといったグリッドコンピューティングでの利用はできない．

グリッドコンピューティングに関して，GLOBUS や UNICORE，NAREGI といったグリッドコンピューティングのためのミドルウェアを利用したものが多数行われている．これらのミドルウェアは，小型計算機を対象としておらず，処理能力不足や記憶容量不足といった理由から，小型計算機には導入できない．グリッドコンピューティングは，研究用途で多数利用されており，センサネットワーク，センサグリッドにおけるセンサノードの観測プロセスや情報取得を記述するための言語として Sensor Model Language (SensorML)[15]がある．SensorML は XML をベースとしており，高い汎用性を持つ反面，人間の読み書きを第一目的としているわけではないため，簡単に利用するにはハードルが高い．

7 まとめ

本研究では，小型計算機を用いたグリッドコンピューティングのためのシステムの要件を整理し，コンパクトグリッドシステムを提案した．コンパクトグリッドシステムは，上記要件を満たすことを考慮したうえで設計，実装されている．また，コンパクトグリッドシステムを構築するために，コンパクトグリッドクライアント，コンパクトグリッドミドルウェア，コンパクトグリッドアセンブラの開発を行った．また，構築した処理連携システムへのジョブ投入のために，処理記述言語を設計し，実装を行った．提案した処理記述言語は手続き型言語として実装を行い，処理連携に必要な設定，大型・小型計算機での処理，結果の出力を簡潔に記述できるようにした．今後，提案した処理記述言語を利用した処理の最適化としてジョブスケジュ

ーリング，ストリームデータの内，音声や映像などの信号を大型計算機で処理することで高品質なメディアデータの放送型配信による提供などについて考えている．

【参考文献】

[1] The Globus Alliance, http://www.globus.org/
[2] UNICORE Forum e.V., http://www.unicore.org/
[3] 国立情報学研究所グリッド研究開発推進拠点, http://www.naregi.org/
[4] T. Yoshihisa, S. Nomura, M. Kanazawa: “A Simple Broadcasting Method for Computation Time Reduction on Grid Computing Environment,” in Proc. of the Int. Workshop on Tools and Applications for Mobile Contents (TAMC’06), pp.68–75, May 2006.
[5] T.Akiyama, S.Shimojo, S.Nishio, Y.Kitatsuji, S.Peltier, T. Hutton, and F.P. Lin: “Telecontrol of Ultra-High Voltage Electron Microscope over Global IPv6 Network,” in Proc. of the Int. Sympo. on Applications and the Internet (SAINT’03) Workshops, pp.184–187, Jan.2003.
[6] 義久智樹: “小型計算機を用いたグリッドコンピューティングのための情報基盤システム,” 並列/分散/協調処理に関するサマーワークショップ(SWoPP2006), 情報処理学会研究報告, 2006-HPC-107, pp.-, July 2006.
[7] TinyOS Community Forum, http://www.tinyos.net/
[8] J. Bentham: “TCP/IP LEAN: Web Servers for Embedded Systems,” SECOND EDITION, CMP Books, 2002.
[9] M.Weiser: “The Computer for the Twentyfirst Century,” Scientific American, Vol. 265, No.3, pp.94–104, Sept.1991.
[10] 早川敬介, 塚本昌彦, 寺田努, 義久智樹, 岸野泰恵, 柏谷篤, 坂根裕, 西尾章治郎: “ユビキタスコンピューティングのためのルールに基づく入出力制御デバイス,” ヒューマンインタフェース学会論文誌, Vol.5, No.3, pp.341-354, Aug.2003.
[11] T. Yoshihisa, Y. Kishino, T. Terada, M. Tsukamoto, R.Sagara, T.Sukenari, D.Taguchi, S.Nishio: “A Rule-Based RFID Tag System Using Ubiquitous Chips,” in Proc.of the IEEE Int. Conf.on Active Media Technology (AMT’05), pp.423–428, May 2005.
[12] Y. Kawahara, M. Minami, S. Saruwatari, H. Morikawa, T. Aoyama: “Challenges and Lessons Learned in Building a Practical Smart Space,” in Proc. of the Annual Int. Conf. on Mobile and Ubiquitous Systems: Networks and Services (MobiQuitous’04), pp.213-222, 2004.
[13] SMART-ITS, http://www.smart-its.org/
[14] MOTE Official Page, http://www.xbow.jp/motemica.html
[15] VAST - Introduction to SensorML, http://vast.uah.edu/SensorML/

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
大型・小型計算機を用いた処理連携システムのための処理記述言語	情報処理学会シンポジウムシリーズ　マルチメディア，分散，協調とモバイルシンポジウム(DICOMO2007) 論文集	2007 年 7 月
A Grid Computing System using Low Resource Compact Computers	Proc. of IEEE Pacific Rim Conference Communications, Computers and Signal Processing (PACRIM’07)	2007 年 8 月