広島工業大学紀要研究編
第42巻（2008）pp. 221-225
論　　文

データ収集システムのためのデータ空間共有型通信フレームワーク

長　坂　康　史*・元　山　裕　基**

（平成19年10月31日受理）

A Shared Dataspace Communication Framework for Data Acquisition System

Yasushi NAGASAKA and Hiroki MOTOYAMA

(Received Oct.31,2007)

Abstract

A shared dataspace communication framework for data acquisition system has been developed. The framework helps us to develop the data acquisition system easily and efficiently. It has been developed on the concept of shared dataspace using a repository. The repository has data and their index keys, and it can be accessed with easy communication commands, PUT, GET, DELETE, NOTIFY and LINK, by the component. The data are managed by the index key in the repository. The component can access the data in the repository by specifying the key related to the data. The new communication commands, NOTIFY and LINK, were also introduced in this framework for a management of the data in the repository. The data acquisition system developer can develop the system easily and efficiently with the shared dataspace communication framework.

Key Words: data acquisition, data management, computer networks

1．はじめに

高エネルギー物理学実験などの測定器実験で使用されるデータ収集システムの効率的な開発は，実験実施における一つの課題である。収集データの大容量化に伴い，データ収集システムの規模と複雑度は増加する傾向にある。また，使用される加速器や測定器の種類，実験の目的や内容によって，データ収集システムに対する要求は様々である。これらのことから，多くのデータ収集システムは，実験ごとの専用システムとして開発されているのが現状であり，開発の効率化が求められている[1][2]。

一方，規模や複雑度が増加しつつあるソフトウェアシステムでは，多数のコンポーネントから構成されるシステムをどのように組織化するかというソフトウェアアーキテクチャの設計に関する問題が重要になってくる。これに関して，アーキテクチャスタイルの有効性が確かめられてきている[3]。アーキテクチャスタイルはシステムが持つアーキテクチャの詳細を隠蔽し，コンポーネント間の相互作用のパターンを明確にする。

そこで本研究では，データ収集システムの開発の効率化を目的として，ネットワーク分散型データ収集システムに共通して必要となるネットワーク通信システムの開発を行うと同時に，データ収集システムに適していると考えられるアーキテクチャスタイルを構築する。アーキテクチャスタイルを構築することで，通信システムに必要となる機能が明確になる。本研究では，その機能を実現する通信システムを開発する。また，アーキテクチャスタイルの観点から通信システムで実現できるコンポーネント間の相互作用のスタイルを明確にすることで，通信システムが適用可能なシステムを判断することができる。

* 広島工業大学情報学部情報工学科
** 日本電気通信システム㈱

2．ソフトウェアアーキテクチャとスタイル

2.1　ソフトウェアアーキテクチャ

ソフトウェアシステムは規模が大きくなるにつれて，どのようにコンポーネントを分割し，分割したコンポーネントをどのように組織化するかといった事が大きな問題になってくる。これはアーキテクチャレベルの問題である。システム全体のアーキテクチャは，その性能やスケーラビリティ，拡張性などの特性に大きく影響を与える。

ソフトウェアアーキテクチャは，アーキテクチャ要素の構成内容によって定義される。アーキテクチャ要素には，コンポーネント，データ，コネクタの３つの種類がある。コンポーネントはインターフェースを介してデータの変換機能を提供する抽象的な構成要素である。データはコンポーネント間で転送される情報のことである。コネクタはコンポーネント間の通信や連携を仲立ちするメカニズムである。コンポーネントはコネクタを介してデータを転送する。

2.2　アーキテクチャスタイル

アーキテクチャにはいくつかのパターンが見られる。様々なアーキテクチャに見られる性質や様式をパターンとしてまとめ，分類したものがアーキテクチャスタイルである。アーキテクチャはアーキテクチャスタイルの具体例と見なすことができるが，アーキテクチャスタイルはアーキテクチャに含まれるシステムの詳細を隠蔽し，コンポーネント間の抽象的な相互作用のパターンを明確にする。これによりアーキテクチャのもつ特性が明確になり，アーキテクチャ同士をその特性によって比較検討することが可能になる。基本的なアーキテクチャスタイルには，パイプ&フィルタスタイルや，クライアント/サーバスタイル，レイヤシステムスタイル，イベントベースインテグレーションスタイルなどがある。

アーキテクチャが要素同士の連携に関する制約を持つのと同様に，アーキテクチャスタイルも要素同士の連携に関する制約を持ち，その制約の集合としてアーキテクチャスタイルして見ることもできる[4]。その場合，スタイルが持つ制約に新しい制約を加えていくことで新しいスタイルを構築するこができる。

3．アーキテクチャスタイルの構築

3.1　データ収集システム

測定器実験で使用されるデータ収集システムの役割は，測定器からの測定データを一ヶ所に集めることである。実験装置内で測定対象となる何らかのイベントが発生すると，システムはそのイベントに関する測定データを測定器から読み取り，そのデータをイベントごとにまとめ，保存する。

ネットワーク分散型データ収集システムでは，システム内でいくつかに分割されたコンポーネントを，ネットワーク上に分散しているノード上に置くことになる。基本的な分散型データ収集システムのノードの配置図を図１に示す。データ収集の主要な処理はデータリーダやイベントビルダ，データレコーダ上で主に実行される。データリーダ・ノード上では，測定イベントが発生すると，測定器からデータを読み取り，イベントビルダ・ノードへ送信する処理が行われる。イベントビルダ・ノード上では，データリーダ・ノードから渡されてくる測定データをイベントごとにまとめ，それをデータレコーダに渡す処理が実行される。データレコーダでは，イベントビルダから受け取ったデータをデータベース等に保存する処理が行なわれる。そして，これらのノードに加えて，分散したノードやコンポーネント間の協調動作を制御するコントローラ・ノードや，システム全体の動作状況等をモニタンリングし，動作環境を管理するマネージャ・ノードが必要となる。

データ収集システムは，システムの開発が完了した後，その実験装置とともに，長期間使用される場合もある。その間に，実験装置の性能向上やデータ量の増加に伴い，システムの処理性能の強化などが行なわれることもある。このような現状において，データ収集システムには機能の拡張性，カスタマイズ性，スケーラビリティなどが要求される。

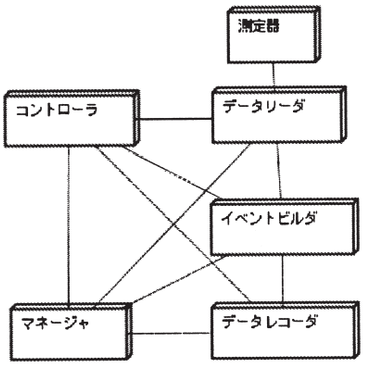

図１　配置図

3.2　アーキテクチャスタイルの構築

本研究で提案するアーキテクチャスタイルはパイプ&フィルタスタイルとイベントベースインテグレーションスタイルを込み合わせたスタイルである。このスタイルの概略図を図２に示す。

パイプ&フィルタスタイルは，フィルタとなるコンポーネントをカスケードに繋げていくスタイルである。データ収集システムが収集しようとするデータは測定器から読み

取られた後，フィルタリングやデータフォーマットの変換，イベントビルド等の処理を受けつつ，最終的にストレージへ保存される。収集されるデータはその過程で，測定器からストレージへ向けた一方向のストリームをシステム内に形成する。本研究のアーキテクチャスタイルはそのようなデータストリーム上にフィルタコンポーネントを設置することでデータ収集を行う。また，パイプ&フィルタスタイルはフィルタの追加や置き換えによるシステムの機能強化や拡張が行いやすい。

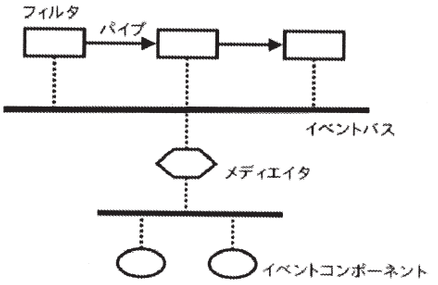

図 2　アーキテクチャスタイル

単純なデータ収集システムでは，パイプ&フィルタスタイルのみに基づいたシステムを構築するだけで十分な場合もあるが，少し複雑度が増すと，コンポーネント同士の協調動作といったパイプ&フィルタには適さない通信が必要となってくる。これに関して，本研究では，イベントベースインテグレーションスタイルを加えることで対応する。イベントベースインテグレーションスタイルは，イベントとプロシージャを関連付け，イベントのアナウンスにより，プロシージャを間接的に起動するスタイルである。イベントはイベントバスによってコンポーネントに配信されるが，コンポーネントはこのイベントバスを介してパイプ&フィルタのデータフローの制御等を行う。また，データ収集の開始や停止といったシステム全体に対する動作の指示や，各コンポーネントやノードの動作状況のモニタリングもこのイベントバスを介して行う。

通常のイベントベースインテーグレーションでは，イベントを配送するイベントバスは１つだけである。そのため，イベントバスに接続するコンポーネントの数に関してスケーラビリティの問題がある。本研究で定めるアーキテクチャスタイルでは，イベントベースインテグレーションスタイルにレイヤーシステムのスタイルを付加させて，イベントバスを階層的に置くことができるようにする。このため，イベントバス同士を仲介するメディエイターコンポーネントがスタイルに追加される。イベントバスを階層的に設置できるようにすることで，ネットワーク通信の効率を高めることができ，スケーラビリティを確保することができる。

以上のように，このアーキテクチャスタイルでは，パイプで接続されたフィルタによってデータ収集を行い，イベントバスを介して，フィルタ間でやり取りされるデータフローの制御，コンポーネントの管理が行われる。このアーキテクチャスタイルに含まれるコネクタ要素は入力ストリームと出力ストリーム，イベントバス，バスコネクタである。入力ストリームはフィルタとなるコンポーネントがデータを受け取るコネクタである。また，出力ストリームはフィルタが処理をした結果を出力するコネクタである。この入力ストリームと出力ストリームを接続することでフィルタ間でのデータの受け渡しが可能になる。また，イベントバスはコンポーネントにイベントを通知する通路になるコネクタである。そして，バスコネクタを使用して，コンポーネントはイベントバスへ接続する。

4．通信システム

4.1　モデル

本研究で提案・開発した通信システムはリポジトリスタイルがベースになっている。リポジトリスタイルでは，データを管理しているリポジトリにコンポーネントがアクセスすることにより，コンポーネント間のインタラクションを実現する。このモデルを図３に示す。

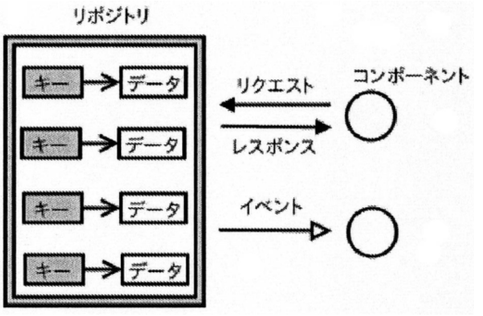

図 3　通信システムのモデル

このモデルでのリポジトリは，キーとデータを関連付けることで，データを管理している。コンポーネントはリクエスト/レスポンス形式の通信をリポジトリと行うことでデータにアクセスする。このとき，リクエストの際にキーを指定することで特定のデータにアクセスすることができる。データへのアクセスにより，データの取得，作成，更新，削除といった操作が可能である。

コンポーネントはリポジトリとのリクエスト/レスポンス形式による通信以外に，イベントと呼ばれるリポジトリからの通知を受けることができる。この仕組みを使うことで，データへのアクセスによりデータの作成や更新が発生すると，その通知を受けられるようにすることができる。

4.2 機能

本研究で開発した通信システムは前述のモデルに基づいた機能を持っている。通信システムのリポジトリが管理しているデータはシリアライズ/デシリアライズ可能な任意のオブジェクトである。このデータを関連付けているキーには文字列を使用している。通信システムが提供する機能は，Get，Put，Delete，Notify，Link の 5 つである。

Get はリポジトリにあるデータを取り出す機能である。Get リクエストをリポジトリに出すことで，この機能を利用できる。キーを指定して Get リクエストをリポジトリへ出すと，そのキーに対応するデータがレスポンスとして返される。キーに対応するデータが無い場合，リクエストの処理はリポジトリ上でブロックし，データが作成されるまでレスポンスは返されない。ただし，Get リクエストにタイムアウト時間を指定すると，ブロックする時間は最長でそのタイムアウト時間までとなる。Put はリポジトリにあるデータを作成・更新する機能である。Put リクエストをリポジトリに出すことでこの機能を利用できる。キーを指定してデータと共に Put リクエストを出すと，指定したキーに対応するデータの作成，または更新が，Put リクエストと共に渡したデータを用いて行われる。

Delete はリポジトリにあるデータを削除する機能である。Delete リクエストを出すことでこの機能を利用できる。キーを指定して Delete リクエストを出すと，リポジトリはそのキーに対応するデータを削除する。キーに対応するデータが無い場合は何もしない。

リポジトリは，データの作成，更新，削除のいずれかが起こると，それをイベントとして捉える。Notify はそのイベントが発生した時，それを通知する機能である。この通知の設定を行うには，リポジトリに Notify リクエストを出す。キーを指定して Notify リクエストを出すと，リポジトリは NotifyID をレスポンスとして返す。リクエストで指定したキーに対応するデータに作成，更新，削除のいずれかが行われると，そのイベントの種類と NotifyID が通知される。また，データの作成か更新によってイベントが発生した場合はそのデータも通知される。

Link は同一リポジトリ，または異なるリポジトリにある 2 つのキーを関連付ける機能である。Link はリンク元キーからリンク先キーへの方向を持ち，リンク元キーに対応するデータの作成，変更，削除があった場合，それと同様の処理がリンク先キーに対応するデータにも行われる。

Link の設定を行うには，リンク元のリポジトリに Link リクエストを出す。このリクエストには，リンク元キーとリンク先のリポジトリ，リンク先キーを指定する。

4.3 コネクタの構築方法

本研究で構築したアーキテクチャスタイルのコネクタ要素は，入力ストリーム，出力ストリーム，イベントバス，バスコネクタである。これらのコネクタは本研究で開発した通信システムを用いて構築することができる。基本的には，各コンポーネントが 1 つずつリポジトリを持ち，お互いが持つリポジトリにアクセスすることでコネクタを構築する。

入力ストリームはリポジトリのキーに対して Notify の設定をするだけで構築できる。Notify の設定をしたキーに対してデータの作成・更新があるとそれが変更内容と共に通知される。変更が発生するごとにその順番を維持して通知が行われ，バッファリングされるため，これは入力ストリームの役割を十分に果たしている。出力ストリームはリポジトリのキーに対して Put リクエストを出すだけで良い。出力ストリームと入力ストリームは互いに接続されることでその役割を果たすことができる。両者の接続には，出力ストリームと入力ストリームを識別するキーを同一リポジトリの同一のキーにする。もしくは，キー同士を Link で接続する。これにより，出力ストリームでの Put によるデータの変更が Link や Notify により伝達され，出力ストリームから入力ストリームへとデータが受け渡される。

イベントバスとバスコネクタはリポジトリそのものがその役割を果たす。イベントバスに対して受信したいシグナルを登録するには，イベントバスになっているリポジトリに Notify の設定をする。このとき指定するキーがシグナルの種類を表している。シグナルを送信するには，送信したいシグナルの種類を表しているキーに対して Put リクエストを送る。Put リクエストによるデータの変更によって，その通知がシグナルを受信したい全てのコンポーネントへ送信され，シグナルの送信が完了する。

4.4 性能評価

開発した通信システムを入出力のストリームとして使用したときのスループットと，イベントバスとして使用したときのシグナル配信の遅延を測定した。

測定には 2 台の PC を用意し，前述したコネクタの構築を行った。測定に使用した PC のスペックを表 1 に示す。入出力ストリームとシグナルで送受信するデータには，単なるビットの塊と整数の配列の 2 種類を使用している。単なるビットの塊では，送受信の際にシリアライズ/デシリアライズが不要であり，システム本来の性能を知ることができる。整数の配列では，シリアライズ/デシリアライズの処理が必要であり，それには要素の数に比例する時間がかかる。この 2 つの測定結果の比較により，データの種類による性能の影響を見る。

表1　マシンスペック

CPU	Pentium III 1GHz
Memory	512MB
NIC	Intel Ethernet Pro 100
Kernel	linux-2.4.21-20.EL.cern

入出力ストリームのスループットの測定結果を図4に，シグナルの遅延を図5に示す。両グラフにあるnetwork performanceの値は通信システムを使用せずに測定した，ネットワークのスループット性能と，ターンアラウンドタイムである。スループットの測定結果を見てみると，単なるビットの塊(raw data)の値はデータサイズ1kB以上では，ネットワークの性能に近いスループットが出ていることがわかる。データ収集システムで収集するデータは1kB未満のサイズのものは少ないことから，スループット上の問題は少ないと考えられる。遅延の測定結果では，raw dataとnetwork performanceを比較すると，データサイズが1 kB以下であるときに遅延の増加が比較的大きく，その差は約1.2msである。イベントバスを流れるデータは1kB以下のものが多いと考えられるが，1.2msの差が大きな問題になるとは考えにくい。

送受信するデータのタイプによる性能の影響を見てみると，データサイズが小さいときにスループットの影響が大きく，また，データサイズが大きくなるほど遅延への影響が大きいことがわかる。このことから，データのフォーマットやシリアライズ/デシリアライズの方法等が性能改善の重要な要素であると考えられる。

5．まとめ

本研究では，データ収集システムの開発の効率化を目的として，ネットワーク分散型データ収集システムに共通して必要になる通信システムの開発を行った。通信システムの開発にあたって，データ収集システム用のアーキテクチャスタイルを構築し，データ収集システムに必要な通信形式を明確にした。その通信形式はパイプとイベント通知である。本研究では，その通信形式をサポートするよう通信システムを開発した。このアーキテクチャスタイルをもつデータ収集システムの開発を行う際，本研究で開発した通信システムを利用することができ，開発の効率化を図ることができると考えられる。また，データ収集システムだけでなく，パイプやイベント通知の通信形式が適しているシステムにこの通信システムを利用することが可能であると考えられる。

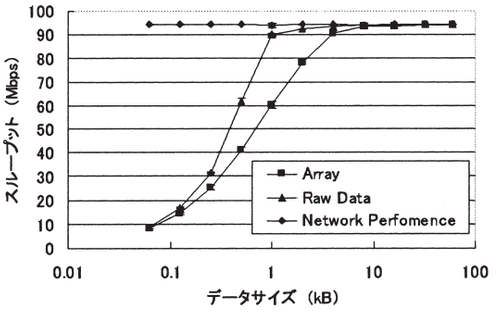

図4　パイプのスループット

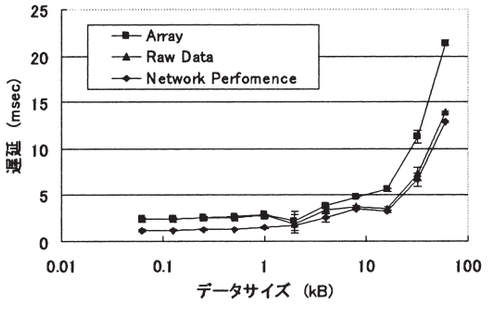

図5　イベントバスの遅延

参　考　文　献

[1] J.Gutleber, et al.: “Towards a homogeneous architecture for high-energy physics data acquisition systems,” Comp. Phys. Comm, Vol. 153, pp.155-163, 2003

[2] Y.Skamoto, et al.: “KONOE: An object- oriented / network-distributed online environment,” IEEE Trans. Nucl. Sci. Vol. 49, pp.3254-3261, 2002

[3] D.Galan and M.Shaw: “An Introduction to Software Architecture,” Advances in Software Engineering & Knowledge Engineering, Vol. 2, pp.1-39, 1993

[4] Roy Thomas Fielding: “Architectural Styles and the Design of Network-based Software Architectures,” PhD thesis, University of California, Irvine, 2000