イベント駆動型分散Javaアプリケーション構築用フレームワークECJ

5M－6

宮澤 隆幸　海邊 裕
（株）東芝　研究開発センター

1. はじめに

Javaは、その適応力の高さから情報系システムの分野で脚光を浴びているが、最近では組込み機器あるいは制御システムへの適用でも注目されている。そこで我々は情報系と制御系のシステムを統合する分散システム構築用フレームワーク ECJ (Event Centric for Java) を開発した。ECJは、情報系システムと制御系システムの統合で要求される、高速な非同期イベント配送を容易に実現可能としており、遠隔監視制御システムをはじめ、家庭内情報システムやオフィス環境の簡易管理サービスなどに適用可能である。本稿ではこのECJの特徴と構成について述べる。

2. 監視制御システムの特徴とECJ

情報系と制御系を統合する分野のアプリケーションの代表的なものに、監視制御システムがあげられる。本節ではこの監視制御システムの特徴と、それに対するECJのアプローチについて述べる。

2.1. イベント駆動型のシステム構築

監視制御システムの多くは、非同期のイベント駆動型システムとして実装される。このためネットワークを介した分散アプリケーションに拡張する場合にも、イベント駆動型システムとして構築することを望むシステム開発者は多い。

ECJはRMIを補完する非同期のイベント配送機能を提供する。これにより監視制御システムの開発者はシステムの設計開発が容易になる。

2.2. 拡張性と保守性の高さ

監視制御システムでは、組込み機器などのリソースに制限のある計算機も使用される。このため、監視制御システムのアプリケーションには実行モジュールのサイズの小ささと、動作の安定性や保守性が要求される。

ECJは通信機構や制御機器に応じたドライバを、それぞれプラグイン可能な構造にしている。さらに、ログの採取機構やエラー状態に関する例外処理を扱う機構を必要時にプラグインすることができる。これにより、拡張性および保守性に優れ、かつ、対象とするシステムに応じた必要最小限の構成を構築することができる。

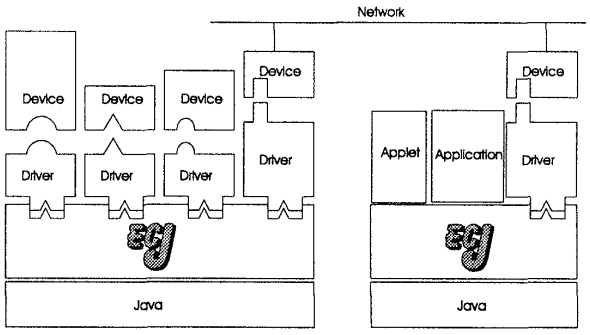

図1　ECJの全体概要

2.3. ネットワーク経由の高速なイベント配送

監視制御システムで扱うイベントは、サイズは小さいが、高速に配送する必要がある。Javaで実装する場合、スレッドのコンテキストの切り替えや、ガベージ・コレクションの実行、例外処理ブロックへの出入りなどが処理速度に大きな影響を与える。

ECJの設計・実装においては、スレッドのコンテキスト切り替えや実行時のオブジェクトのコピー、例外処理ブロックへの出入りを最小化することで処理の最適化を図った。これによりECJのイベント配送速度は、ネットワークを経由した場合でも数ミリ秒以下となった。この値は実行環境に依存するが、C言語などで実装する場合と同等の性能である。

2.4. 既存システムの統合

ECJでは既存の監視制御システムを容易に統合するため、イベントの統合モデルをサポートしている。このモデルは、既存のシステムで独自に規定されたイベント群を抽象化し、これらを継承関係とし

ECJ: A Framework for Event Driven Distributed Java Applications
Takayuki Miyazawa, Hiroshi Kaibe
Toshiba Corporation, Research and Development Center

て体系化するものである。

これにより既存システムのイベントを任意の型に変換することが可能となり、既存の監視制御システムを容易に統合することができる。また、ネットワークなどへのアクセスを提供するドライバ層も抽象化して扱っているため、通信層のプロトコル変更などにも容易に対応できる。

2.5. プラットホーム独立

ECJはすべてJava言語で書かれており、プラットホームに依存するライブラリを使用していない。このため、Java環境の動作するプラットホームであればどこでも使用することができる。

3. ECJの機能構成

ECJの機能構成図を次に示す。

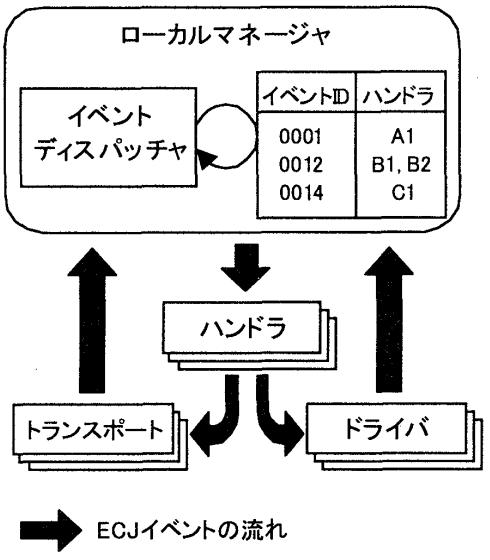

図2 ECJの機能構成

以下でそれぞれについて述べる。

3.1. ECJイベント

ECJが扱うイベントである。これは抽象化を行ったイベントであり、イベントIDとデータから構成される。

イベントIDは64ビットの空間を持つ。データは、イベントIDごとに任意の利用方法が許されている。

3.2. ローカルマネージャ

ECJの管理部である。内部にイベント ディスパッチャとイベント テーブルを持つ。イベント テーブルはイベントIDとイベントハンドラの対応表である。この対応表に基づき、イベント ディスパッチャは各ドライバやトランスポートからのECJイベントを適切なハンドラに引き渡す。

3.3. ハンドラ

ローカルマネージャによってECJイベントのIDに応じて起動される処理手続きである。このハンドラは抽象クラスとして定義されており、それを継承してそれぞれのECJイベント特有の処理ロジックを記述する。

3.4. ドライバ

既存の制御システムなどで独自に規定されたイベント群をECJイベントに変換する、あるいはその逆変換を行うドライバである。ドライバは抽象クラスとして定義されており、この実装は制御システムのベンダあるいはドライバのサードパーティ ベンダが供給する。

また、ECJの実行系は実行時にドライバを動的に追加あるいは削除することを許している。このため、ユーザによるドライバの選択や変更が可能となり、システムの動的な統合あるいは変更が容易になる。

3.5. トランスポート

ECJイベントをネットワーク経由で配送する機構を提供する。すなわち、異なる計算機上で実行されるECJの実行系間のイベント通信路として機能する。

このトランスポートは、前述のドライバの一種として定義される。このため、ユーザは通信層のプロトコル選択などをドライバと同様に容易に行うことができる。

4. おわりに

本稿では、イベント駆動型分散Javaアプリケーション構築用フレームワークECJの概要について述べた。現在、ECJの試用版を東芝WWWサーバにて公開している。興味のある方はご利用いただきたい。URLは、http://www2.toshiba.co.jp/ecj/ である。

参考文献

[1] 江口 他, "JavaによるJavaのためのイベント配送フレームワークECJ", 東芝レビュー Vol.53 No.8, pp.7-10, 1998

[2] Object Management Group Inc., "CORBA Services: Common Object Services Specification", 1997