Common Logging Interface
Ein System zum Sammeln und Weiterverarbeiten von Debugnachrichten in verteilten Umgebungen

Raimar Falke **Michael Peter** **Achim Gratz**

Rainer G. Spallek

Institut für Technische Informatik

Technische Universität Dresden

01062 Dresden

E-mail: {falke,peter,gratz,rgs}@ite.inf.tu-dresden.de

WWW: http://www.inf.tu-dresden.de/TU/Informatik/TeI/

13. November 1998

Gliederung

Einführung

- Motivation
- Anforderungen
- Architektur

Filterung

- Grundlegendes
- Realisierung
- Beispiel
- Erweiterungen

Ausblick

- Performance
- Bedienoberfläche

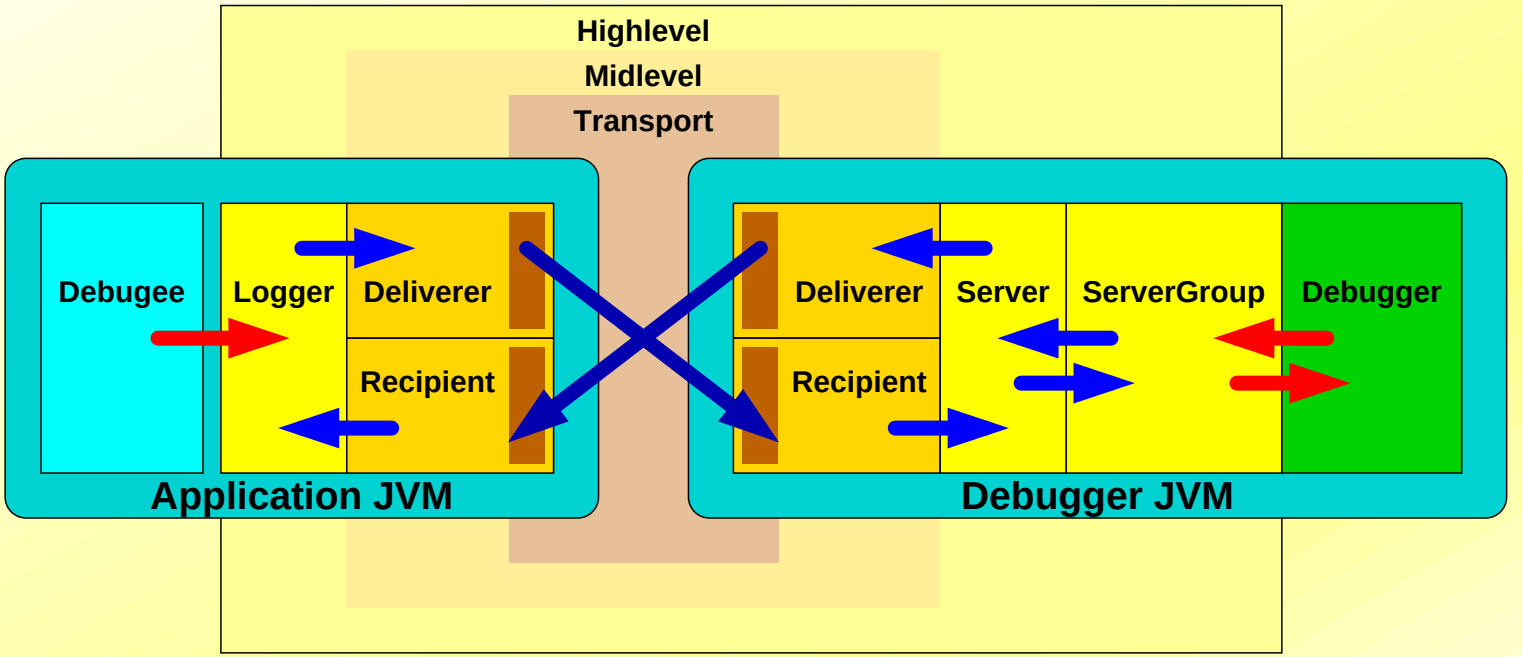
Anforderungen von der Benutzerseite

- Unterstützung für verteilte Anwendungen
- Unterstützung für MT-Programme
- einfaches API für den Anwender
- leicht konfigurierbar und vollständig abschaltbar
- Filterung von Nachrichten anhand von zur Laufzeit vorgebbaren Kriterien
- zeitnahe und möglichst zeitrichtige Beobachtung der Nachrichten
- Protokollierung der Nachrichten

Resultierende Anforderungen an die Architektur

- Client-Server (Multi-Client, eventuell Multi-Server)
- Flexibilität durch Aufbau von Schichten
- lokal und im Netzwerk benutzbar
- reentrante Aufrufe durch MT-Client-Programme
- Pufferung zur Minimierung von Latenzen
- Toleranz gegenüber Fehlern in den Client-Programmen und im Netzwerk
- Skalierbarkeit, Effizienz und Zuverlässigkeit

Architektur von CLI



Filterung - Grundlegendes

- die meisten Nachrichten in einem gegebenen Zeitraum sind irrelevant
- die Relevanz der Nachrichten wird vom Beobachter festgelegt
- das Relevanzkriterium ist nicht statisch
- verschiedene Teile des Systems befinden sich in unterschiedlichen Zuständen

Nur die relevanten Nachrichten sollen zum Beobachter durchgelassen werden!

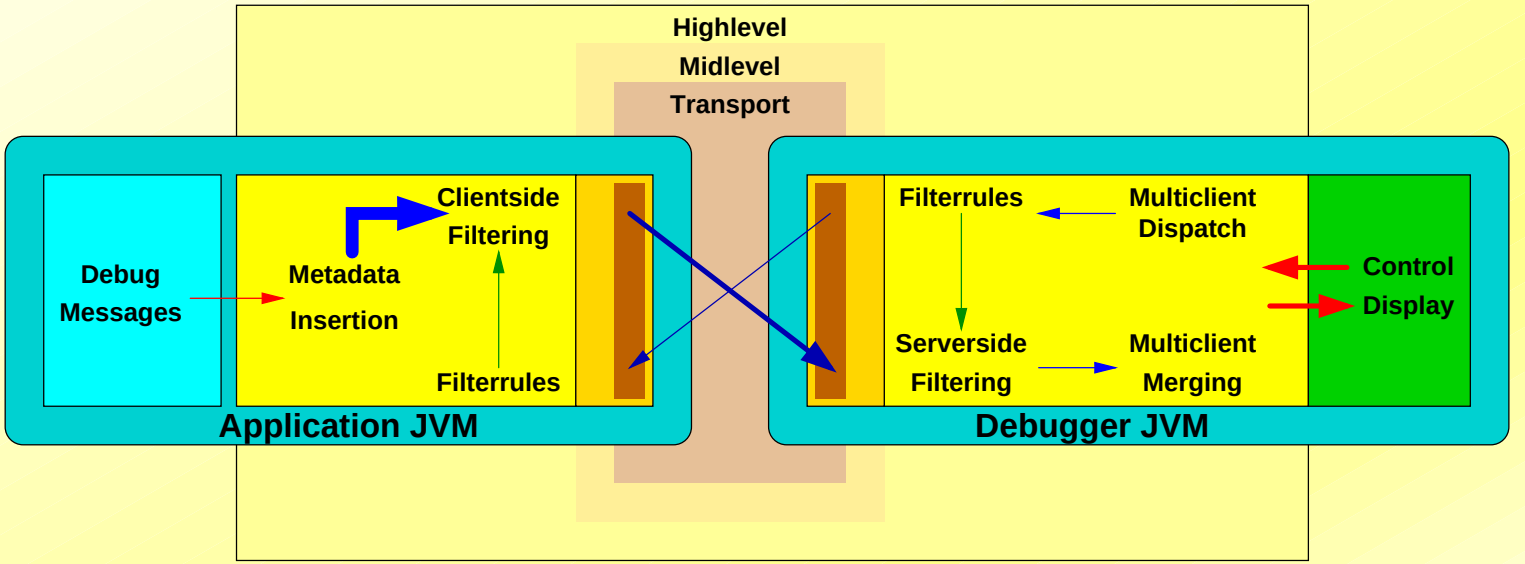
Filterung - Realisierung

- Datenaufkommen möglichst nahe der Quelle reduzieren
- weitere Selektion am Beobachtungsort

Die Filterung muß mindestens zweistufig erfolgen!

- Vorfilterung im Client
- Nachfilterung im Server

Filterung - Datenfluß



Filterung - Kriterien

Nachricht

- Zeichenketten

Typ

- benutzerdefinierte Klassifikation

Metadaten

- Threadgroup, Thread
- Package, Klasse
- Instanz
- VM
- Host

Filterung - Beispiele

- Filterregeln des Clients

```
TypeName!="Debug" +forward
ThreadName|="Runner";ClassName="Observer"\
 +saveLong "observer.log"
```

- Filterregeln des Servers

```
HostName="a.b.com";Msg|="(can not|can't).*socket"\
 +saveLong "socketproblems.log"
TypeName="Error" +displayShort
```

Filterung - Erweiterungen

- mehr als zweistufige Filterung (Kaskadierung von Servern)
- mehr als ein Server pro Client (konkurrente Beobachtung)
- verbesserte Algorithmen zur Filterung
- Verwalten der Filterregeln durch GUI
- Reaktion auf Ereignisse und Roll-Back/Roll-Forward

Filterung - Ergebnisse

- das Filterkonzept ist eine tragfähige Basis
- Server und Netzwerk werden signifikant entlastet
- zusätzliche Belastung des Client kann im Normalfall vernachlässigt werden
- weitere Verbesserungen des Konzeptes und der Implementierung sind angedacht

Zusammenfassung

- das Grundkonzept hat sich bewährt
- die Performance ist für erste Versuche ausreichend
- höhere Flexibilität teilweise auf Kosten der Performance erzielt
- Fehlertoleranz ist noch nicht zufriedenstellend
- Skalierbarkeit ist in der vorliegenden Implementierung begrenzt
- Qualität der JVM-Implementierungen verbessert sich kontinuierlich

Ausblick

- Erweiterungen des Konzeptes und der Implementierung
- Verbesserung der Fehlertoleranz und Skalierbarkeit
- Bedienung und Konfiguration mittels einer grafischen Oberfläche
- Verbesserung der Implementierung
- Lokalisierung von Flaschenhälsen