

Verteilte Systeme

April 28, 2015

1 ØMQ

1.1 Verwendung

1. Context: Es muss für ØMQ ein Kontext erstellt werden (das Umfeld)
2. Socket: Es muss ein Socket in diesem Kontext erstellt werden
3. Sockettyp: Was für eine Art Socket wird benötigt (SUB/PUB etc.)
4. Binding: Socket an eine Adresse und einen Port binden (Remote, oder Lokale für listening)
- 5.1 Receive Receive (blockierend), besonderer Messagetyp
- 5.2 Send Send, gegebener Messagetyp ist zu füllen

ØMQ Buffert alle Nachrichten bis zu einem gewissen Grad. So werden auch langsamere Reader oder später registrierte Subscriber bedient, jedoch ohne versicherung, dass sie alle Nachrichten empfangen.

1.2 Patterns

Request-Reply Synchroner Request-Reply Kommunikation

Push-Pull Sender erwartet keine Rückantwort (Push), Empfänger benutzen Pull, um Nachricht zu empfangen. Kann benutzt werden, um Lasten zu verteilen, weil eine Nachricht nur höchstens einmal empfangen werden kann.

Pub/Sub Ein Publisher sendet Nachrichten mit bestimmten Tags, alle Subscriber empfangen alle Nachrichten, die den vorgegebenen Tags des Subscribers entsprechen.

1.3 Transportvarianten

ipc: Interprozess, FIFO etc.

inproc: Socketkommunikation innerhalb eines Prozesses.

mMulticast: ØMQ unterstützt Multicast, wenn das Netzwerk es unterstützt.

1.4 PlayØMQ, Broker

Sources generieren Zufallszahlen und geben sie an den Broker.

Worker nehmen Zahlen aus dem Broker und prüfen auf Primzahl.

Sinks hören auf Nummern/Primzahlen. Broker: Sendet einkommende Zahlen per Push-Socket an Worker, sendet Zahlen und Primzahlen per PUB-Socket Source: Request Reply auf Broker.

Worker: Push/Pull → Parallelisierung der Berechnung

Sink: Subscriber, die auf den Broker "Subscriben", mit dem entsprechenden Tag (number/primzahl)

1.4.1 Nachrichten

Senden und Empfangen von Daten muss besonders behandelt werden, da Nachrichten erstellt werden müssen. Nachrichten bestehen aus Tag (identifizier Char) und Wert (long)

1.4.2 Source

Benötigt URL von Broker

Erstelle REQ (Request) Socket, der mit dem Broker verbindet.

Source generiert Zahlen, sendet sie an Broker und wartet, bis dieser den Empfang bestätigt hat.

1.4.3 Broker

Broker hat Sockets für REP (Reply) für Source, PUB (Publisher) für Sinks, PUSH für Worker.

Im Beispiel ist Endpoint eine Methode, die die Befehle zum Erstellen eines einfachen Sockets zusammenfasst.

Broker empfängt Zahl von Source, bestätigt dem Empfang, publiziert die Nummer, sendet sie an die Worker, wartet auf die Antwort und publiziert u.U. die Primzahl.

1.4.4 Worker

Verbindet per PULL und REQ (Request) Socket auf den Broker.

Empfängt vom Broker per PULL Zahlen, prüft diese und sendet sie u.U. per REQ an Broker. Wenn die Primzahl gesendet wird, muss der Worker wieder auf die Antwort (REPLY) des Broker warten.

1.4.5 Sink

Erstellt SUB (Subscriber) Socket, über den er vom Broker Daten empfängt.

Beim Erstellen wird festgelegt, auf welche Tags der Subscriber hört.

2 Verteilung von Last

Kreative Lastverteilung: Verteilte Programmierung.

Mechanische Lastverteilung: Verteilung auf Rechner, gleiche Auslastung der verwendeten Rechner

Durch verteilte Rechner ist die "aktuelle Last" verzögert, um die Latenz zwischen A und B. A schickt also Last an B, auch wenn B inzwischen eine hohe Last erreicht hat.

Lösung: A kann anhand der verteilten Last merken, wie viel Last ungefähr an B vergeben wurde (wenn B Last nicht weitergibt).

Paketweitergabe:

- Wenn der Code lokal vorhanden ist, muss er kopiert werden (achte auf identische Pfadnamen etc.)
Bei Netzwerklaufwerken geht Zeit verloren, da das Netzwerklaufwerk limitierende Bandbreite hat.
- Verteiler schickt Programmcode mit (Große Datenmenge, Achtung Viren)

2.1 Architektur

Erstellen einer Lastmetrik: Welche Last hat ein Rechner, welche Last erzeugt ein Paket?

Verteilung der Last der beteiligten Rechner: Entscheidungsgrundlage für Verteiler.

Verteilung des Lastpaketes auf gewählte Rechner: Welche Informationen müssen übertragen werden.

Lastmetriken

- Prozessorauslastung (Anzahl Prozesse, CPU-Zeit)
- Speicherauslastung (Bedarf etc.)
- Kommunikationslast (IO, Netzwerk)

Pull

Zieht Lastwerten an. (Unterforderte Rechner suchen Last)

Lastwert wird bei der Erzeugung eines Paketes ermittelt.

Varianten:

n allen Rechnern: Verteiler sendet Broadcast und fragt nach Last, alle Rechner antworten mit eigener Last.

Verbesserung: Antwort wird verzögert, je nach eigener Last. Damit antwortet der am wenigsten ausgelastete als erstes.

Feste Teilmenge: Errichte bestimmte Worker

zufällige Teilmenge: Frage zufällige Worker an

Push

Lastwerte verteilen. (Überforderte Rechner verteilen Last)

2.2 Verteilungsverfahren

Last sollte so verteilt werden, dass Rechen- und Speicherlast gleich verteilt ist und die Kommunikation möglichst lokal stattfindet.

statisches Verfahren Optimale Verteilung vor dem Start der Anwendung ermitteln.

Alle Bestandteile der Lastmetrik sind für alle Prozesse bekannt.

Findet häufig Anwendung, Last ist vorher meistens bekannt (Meteorologie etc.)

dynamisches Verfahren Ausführungsort wird für jeden Prozess ermittelt.

Mit Migration Migration heißt, die Adressräume und die Prozessstate zu übertragen. Es muss gewartet werden, bis alle Threads keine OS-Ressourcen mehr benutzen.

Problem: Adressräume sind u.U. sehr groß.

Lösung: Copy-on-Reference: Kopiere Adressen, wenn sie benötigt werden. (Reduziert Speicherlast auf Quellknoten nicht).