# 基于单元的分布式计算模型 CalUnit

## CalUnit Unit – Oriented general distributed computing platform

刘 盛 邵 栋 郑 滔 (南京大学软件学院 210089)

**摘要**:本文在 JUnit 单元测试技术的基础上,提出基于单元的分布式计算模型 CalUnit。对 CalUnit 设计模型及实现方法进行了较详细的论述和分析,系统测试表明其计算速度提升十分可观。该模型可作为一种实用型分布式计算框架结构,并扩展分布式计算的应用领域。

**关键词**:分布式计算 科学计算 单元化 JUnit

本文提出的 CalUnit 是参照 JUnit 单元模型的一种新的分布式计算平台。它采用 C/S(Client/Server 客户机/服务器)工作模式,并使用网格分布式计算概念,将所有的计算终端(遍布于 Internet 上的各种计算设备)通过中心服务器进行在线连接,中心服务器接受大量统一的计算任务单元,并实时向终端计算设备发放计算任务包,然后负责回收计算结果,最终生成计算结果报告。

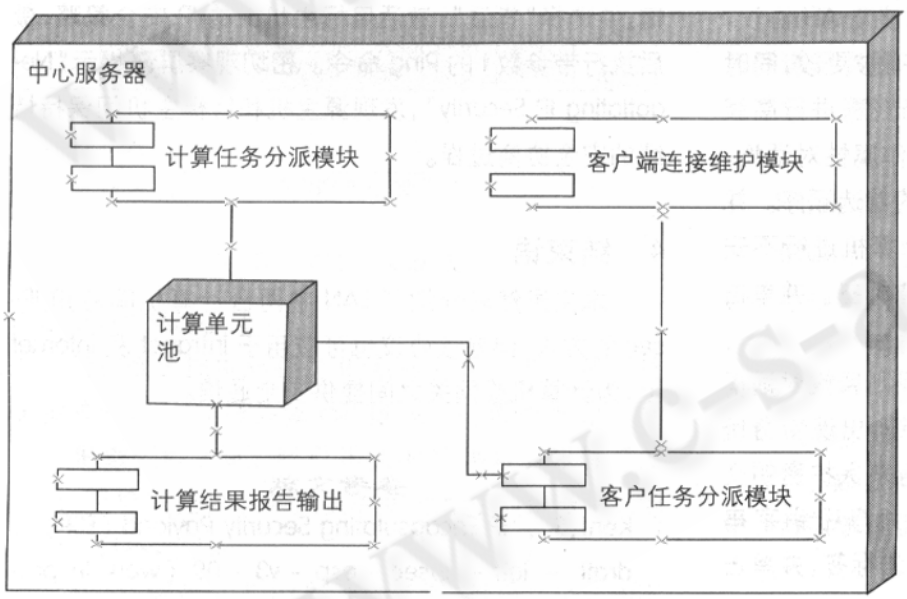


图 1 CalUnit 设计模型结构示意图

## 1 CalUnit 设计模型

CalUnit 的设计模型所描述的是 Intranet 内联网分布式计算模型,共分为任务分派模块、计算单元池、客户端连接维护模块、客户任务分派模块四个功能模块。CalUnit 设计模型结构见图 1。

任务分派模块将用户提交的按统一接口格式编写的计算任务,根据任务的要求拆分成为许多计算单元,压入中心服务器计算单元池中。

在计算单元池模块中,每个计算单元按规定的逻辑划分成为计算数据和计算描述两个部分,并使用 Command 模式将所有的任务管理编成一个命令管道队列,以便于任务分派模块从队列中顺序提取计算单元任务。

客户端连接维护模块在固定端口(本文给的例子使用 6701)监听所有请求连接或连接在线的客户端,并负责维护连接及处理连接的断开。同时,客户连接模块还必须维护每个客户端当前的状态(Suspend; Waiting; Ready; Error),并能够让其他模块及时得到相应的状态信息。

客户任务分派模块通过预先制定的选择逻辑或决策逻辑,根据客户端所处状态,连接计算单元池与客户端连接维护模块,取出某一计算单元并发送给处于就绪状态的客户端。

## 2 技术细节讨论

### 2.1 计算任务的描述与交换

计算任务的标准化描述与交换是 CalUnit 分布式计

算平台的关键技术，在 CalUnit 中计算任务被划分成为计算数据和计算描述两个部分。计算数据是一个用户负责序列化的对象（实现 Serializable 接口的对象），计算描述是一个包含有算法描述的类，利用 Java 远程类引用技术将类发送到客户端（该类实现了 Calculate 接口）：

```
public interface Calulate{
    public String process(String input);
}
```

计算数据可以通过 Socket 流进行发送和接收。运算模块通过 J2SE 提供的远程类的 ClassLoader 实现：

```
protected byte [ ] getClassData ( String className )
throws IOException {
    byte[ ]data;
    int length;
      try {
        URL url = new URL(className.endsWith(".
class") ?
          className : className + ".class");
        URLConnection connection = url.openCon-
nection();
        InputStream inputStream = connection.getIn-
putStream();
        length = connection.getContentLength();
        data = new byte[length];
        inputStream.read(data);
        inputStream.close();
        return data;
      } catch(Exception e) {
        throw new IOException(className);
      }
    }
```

### 2.2 网络连接维护

网络连接采用 Socket 网络连接技术。由于计算任务传输的数据已经做到了序列化（Serialized），因此，使用普通的 TCP/Socket 连接就可以达到数据传输的要求，并且网络利用效率能够达到最大。这样的设计可以满足计算中心内联网网络设备单一的情况下网络分布式计算问题。

如果考虑 Internet 广域网的情况就要复杂一些，Socket 不能解决诸如 NAT 等问题。而采用基于 Web Service 的模式就能方便地解决此类问题。本文给出的验证实例采用 Socket 进行网络连接：

```
ServerSocket s = new ServerSocket(PORT);
try{
  while(true){
    Socket socket = s.accept();
    Runner client = new Runner(socket,pr);
    client.start();
    System.out.println("Accepted " +s);
  }
}
finally{
  s.close();
}
```

### 2.3 计算任务的形式化表示

由于有些科学计算数据对应用软件具有相当强的依赖性，例如大型迭代方程的求解等，这样的计算任务暂时不宜作为基于单元的 CalUnit 分布式计算问题处理。但正如 JUnit 将面向单元测试一样，CalUnit 模型将能适合下述特定类型的科学计算。

（1）计算输入数据非常庞大的问题。需要对每块特定数据进行特定的处理，使每部分数据之间相对独立，例如著名的 Internet 分布式项目：在家寻找外星人计划 SETI@ home。这样的计算任务在许多以实验为主的实验科学领域里大量存在，CalUnit 分布式计算技术可大大的提高其计算效率。

（2）计算工作量十分巨大的问题。只要其计算任务比较容易分解成独立的部分。采用 CalUnit 分布式计算技术就可以提高其计算速度，例如计算 π 的近似值，寻找梅森质数等，本例中给出的一个简单的大型计算任务：计算 100000！也属于此类。

（3）单元的概念还可以应用于一些非计算领域，实现一些其他资源的分布式共享，比如搜索引擎算法、P2P 中的端口发现等。

## 3 CalUnit 分布式计算实例

### 3.1 模型概述

为了验证以上构想和计算效率，设计了一个简单的分布式单元计算框架模型 MyCalUnit，并以纯数学计算 100000 的阶乘为实例构建一个简单的大型计算任

务。

正如前面所描述的设计模型，本实例也分为任务分派、计算单元池、客户端连接维护、客户任务分派等四大块工作。由于该设计为一个模型系统，没有包含一个独立的结果报告输出模块，直接使用控制台的输出形式来输出结果。

模型系统使用的编译器和运行环境为 J2SE 1.4.

### 3.2 计算算法设计

由于 100000 阶乘的计算结果数值过大，采用一个 Integer 型的数组来描述整个结果，每一个数值单元（本例中使用 5 个十进制位数）用一个 int 变量来表示。因此将每一个单元设计为数组中的一段整数值，并动态的将这一段整数分派给某个客户端进行计算。计算的结果最终在 Console 上以文本文件形式输出，以便检查其正确性。

### 3.3 计算时间测试结果

MyCulUnit 计算耗时结果如图 2 所示。

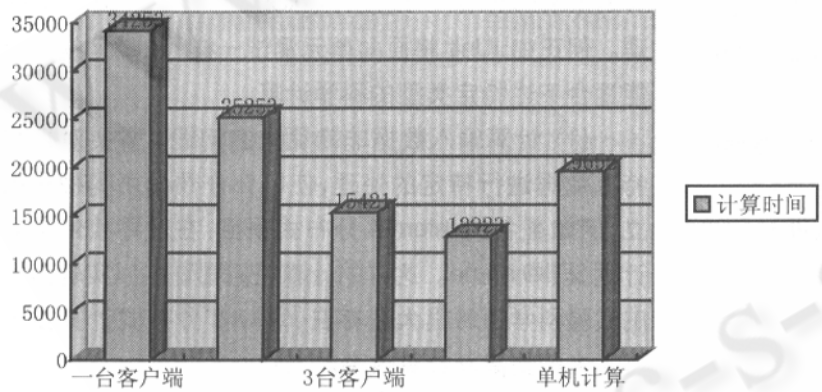


图 2　计算耗时结果

从实例的测试结果可以看到，当一台或两台客户机连接时，计算时间仍然大于单机计算，而当客户端数目超过 3 台之后，计算时间就越来越短于单机计算时间。尽管 Internet 上的网络环境不如测试环境，但计算设备也会远远多于 4 台，因此该实例的效率在 Internet 上仍然是可以接受的。

## 4　系统的特点

(1) 任务的设计、交流和传递方便。系统的最大特点在于用单元化、标准化的思想处理分布式计算的问题，方便用户设计计算任务，便于任务的交流和传递。

(2) 跨平台性能好。系统采用 Java 语言来开发。因此，不论是平台系统还是计算任务本身都具有十分优秀的跨平台性能。

(3) 计算效率高。理论分析与实际验证都证明，该系统具有良好的计算效率，能够解决工程应用中多种类型的大型科学计算问题。

## 5　存在的问题

(1) 模型系统使用的网络连接方式为 TCP/Socket，在 Internet 异构网络环境下直接使用是行不通的，实际的实现方案应根据具体情况采用不同的网络协议与网络连接方式。

(2) 网络传输带来的时间消耗仍然很高，以至于在客户端增多的情况下，效率的提升速度受到限制，根据实际情况可以估算出模型系统计算效率的上限。

(3) 模型系统还不能同时处理多个计算任务。

## 6　结论

基于单元的 CalUnit 分布式计算模型，可以将原本繁杂的分布式计算进行单元化，使大型计算在遵循统一接口标准的条件下，实现计算任务跨系统、跨平台的分配与交流，从而可最大限度的利用网上资源资源，提高计算速度和计算效率。同时，可以拓宽分布式计算技术的应用领域，加速其推广应用。

## 参考文献

1 Nancy. A. Lynch, Distributed Algorithms, Morgan Kanfmann Publishers, 1996.
2 Bruce Eckel, Thinking in Java, Prentice – Hall, 2002.
3 Eric Armstrong & Stephanie Bodoff, The Java Web Service Tutorial, Addison – Wesley's 2002.
4 Glen Broce& Rob Dempsey, Security in Distributed Computing, Prentice Hall, 1997.