

第1章 分布式计算概述

提 纲

- 分布式计算概念
- 分布式系统介绍
- 分布式计算基础技术



分布式计算的定义

分布式计算是一门计算机科学，主要研究对象是分布式系统。在介绍分布式计算概念前，首先简单了解一下什么是分布式系统。简单地说，一个分布式系统是由若干通过网络互联的计算机组成的软硬件系统[1]，且这些计算机互相配合以完成一个共同的目标（往往这个共同的目标称为“项目”）

分布式计算指在分布式系统上执行的计算。分布式计算是将一个大型计算任务分成很多部分分别交给其他的计算机处理，并将所有的计算结果合并为原问题的解决方案。这里与并行计算不同的是，并行计算是使用多个处理器并行执行单个计算。

分布式计算的优缺点

优点

- 超大规模
- 虚拟化
- 高可靠性
- 通用性
- 高可伸缩性
- 按需服务
- 极其廉价
- 容错性

弱点

➤ 多点故障

一台或多台计算机的故障，或一条或多条网络链路的故障，都会导致分布式系统出现问题

➤ 安全性

分布式系统为非授权用户的攻击提供了更多机会

分布式云计算相关计算形式

分布式计算
搜索字词

云计算
搜索字词

物联网
搜索字词

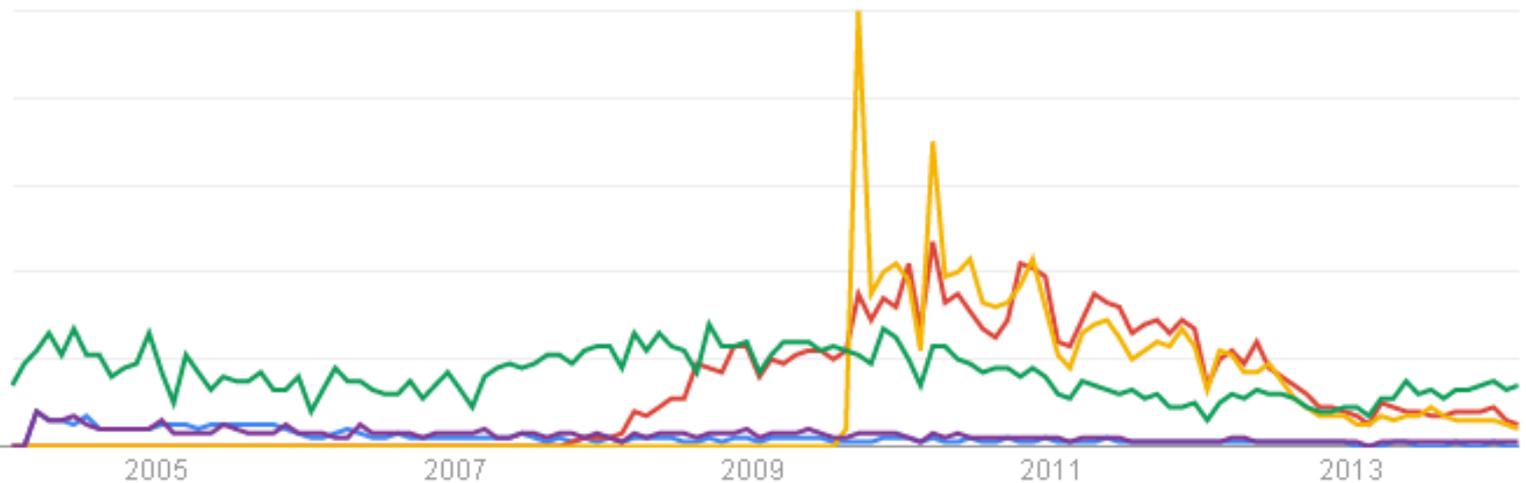
大数据
搜索字词

并行计算
搜索字词

分享 ▾

热度随时间变化的趋势 ?

新闻头条 ? 预测 ?



分布式云计算相关计算形式

distribute...

搜索字词

cloud co...

搜索字词

internet ...

搜索字词

big data

搜索字词

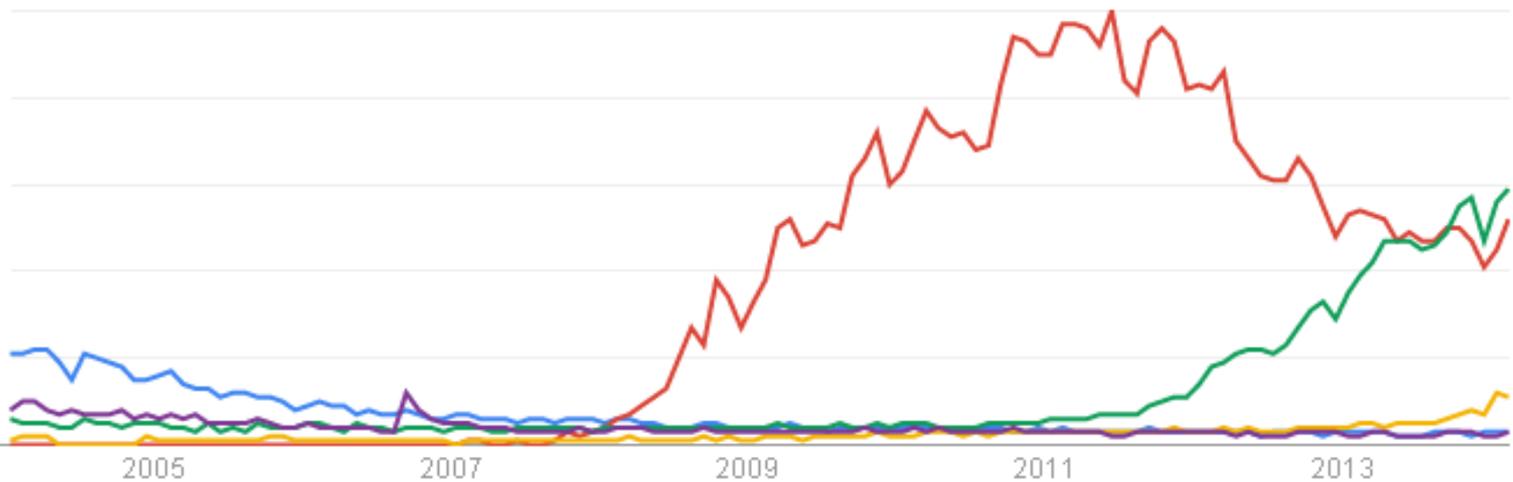
parallel c...

搜索字词

分享 ▾

热度随时间变化的趋势 ?

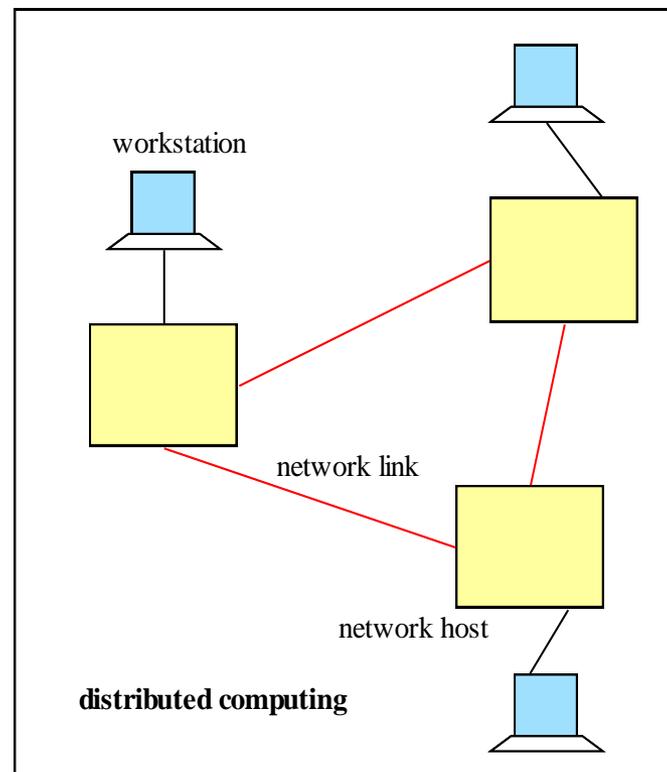
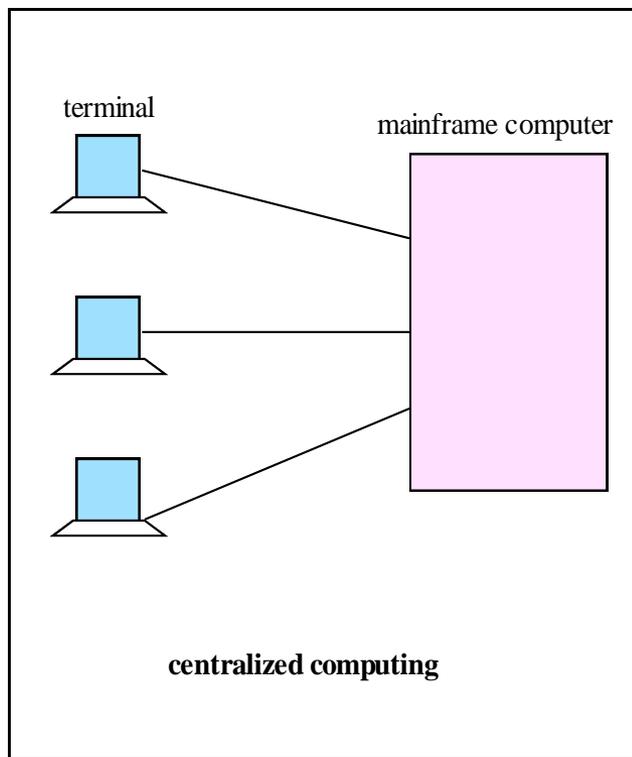
新闻头条 预测 ?



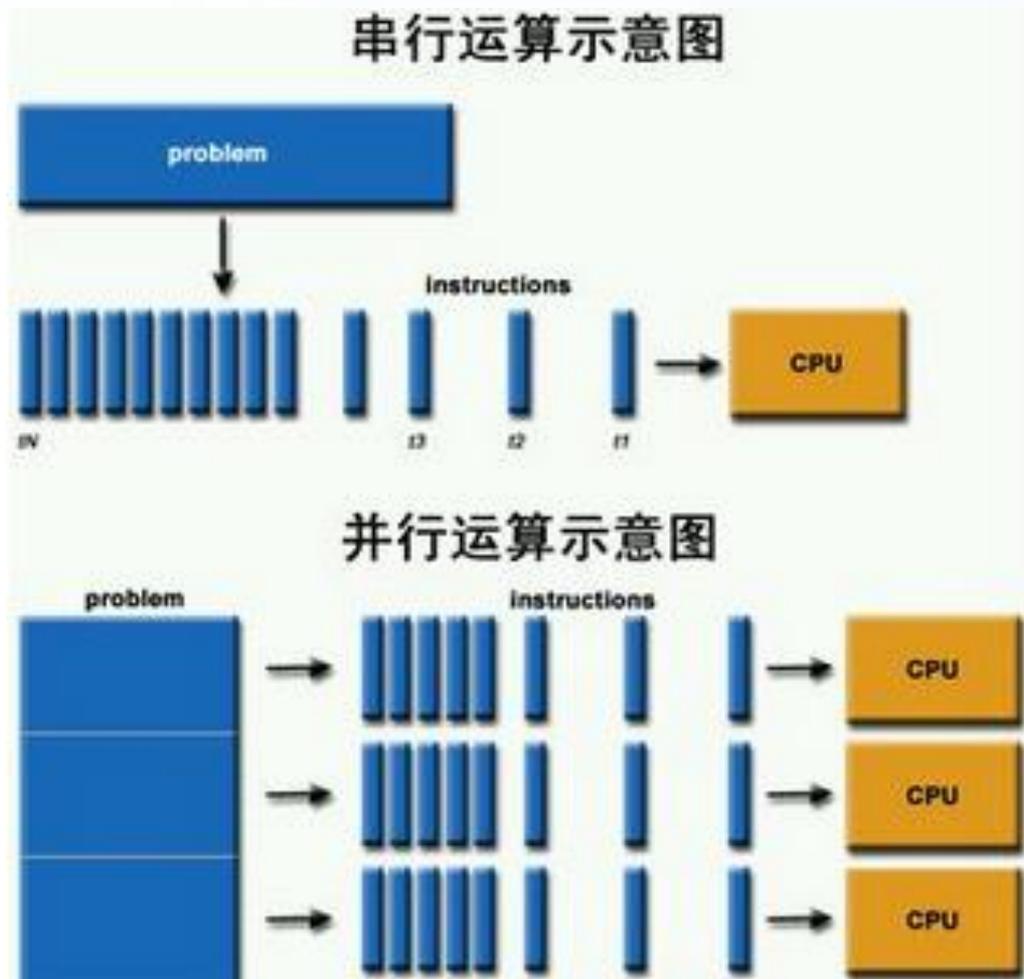
平均值



集中计算与分布式计算



串行运算与并行运算

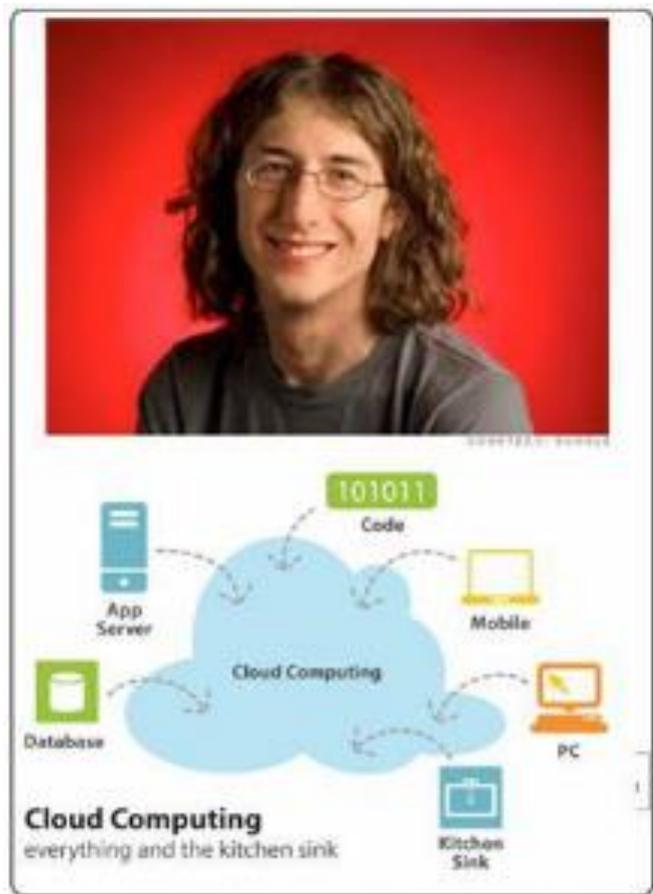


并行运算与分布式计算的区别是：分布式计算强调的是任务的分布执行，而并行计算强调的是任务的并发执行

分布式计算·云计算·大数据

云计算概念提出

- 提出者：Google工程师，比希利亚，27岁
- 出发点：推广Google超级强大的计算资源，招贤纳士
- 结果：蜂拥而至，名噪一时，被视为“云”的起源

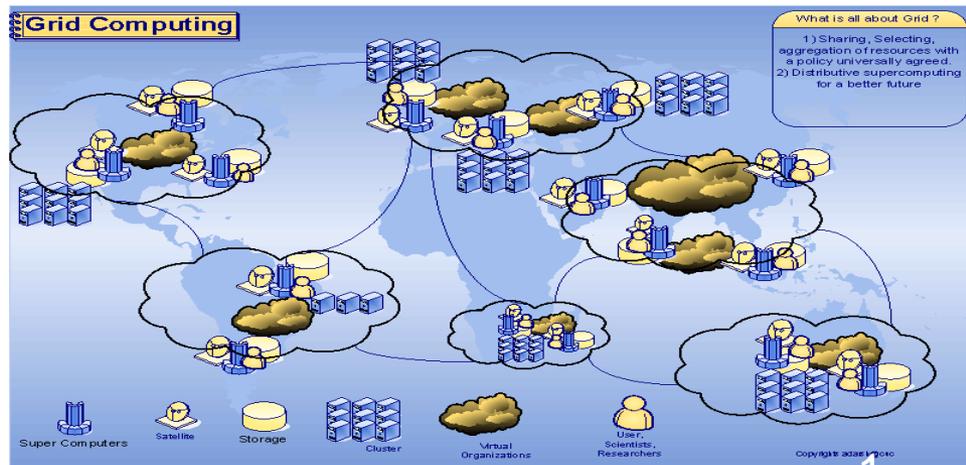


云计算与网格计算

- ❖ **网格计算**：利用互联网把地理上广泛分布的各种资源（计算、存储、带宽、软件、数据、信息、知识等）连成一个逻辑整体，就像一台超级计算机一样，为用户提供一体化信息和应用服务（计算、存储、访问等）。
- ❖ 网格计算强调资源共享，任何节点都可以请求使用其它节点的资源，任何节点都需要贡献一定资源给其他节点。云计算强调专有，请求或获取的资源是专有的，并且由少数团体提供，使用者不需要贡献自己的资源。
- ❖ 网格计算侧重并行的计算集中性需求，并且难以自动扩展。云计算侧重事务性应用，大量的单独的请求，可以实现自动或半自动的扩展。

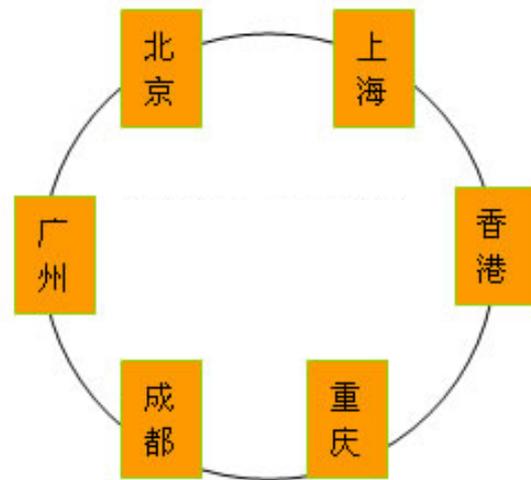
云计算包含的网格计算特征：

- (1) 提供在线的计算、存储等服务
- (2) 超大规模的资源组合
- (3) 资源的虚拟化



云计算与分布式计算

- ❖ **分布式计算（狭义）**：将待解决问题分成多个小问题，再分配给许多计算系统处理，最后将处理结果加以综合。
- ❖ **特点**：把计算任务分派给网络中的多台独立的机器
- ❖ **优点**
 - 稀有资源可以共享
 - 通过分布式计算可以在多台计算机上平衡计算负载
 - 可以把程序放在最适合运行它的计算机上



云计算包含的分布式计算特征：

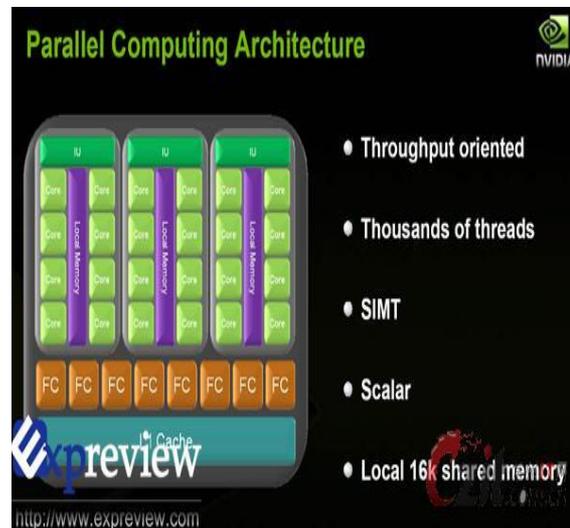
- (1) 通过资源调度和组合满足用户的资源请求
- (2) 对外提供统一的、单一的接口

云计算与并行计算

- 并行计算：是指同时使用多种计算资源解决计算问题的过程。通常指一个程序的多个部分同时运行于多个处理器上。
- 特点：把计算任务分派给系统内的多个运算单元
- 并行计算问题的特征
 - 将工作分离成离散部分，有助于同时解决
 - 随时并及时地执行多个程序指令（多条线同时运行）
 - 多计算资源下解决问题的耗时要少于单个计算资源下的耗时

云计算包含的并行计算特征：

(1) 用户资源（单一类型和组合类型）请求的同时处理



云计算与对等计算

- 对等计算系统中，每个节点都拥有对等的功能与责任，既可以充当服务器向其他节点提供数据或服务，又可以作为客户机享用其他节点提供的数据或服务，节点之间的交互可以是直接对等的，任何节点可以随时自由地加入或离开系统。

对等计算：有可能作为云计算的一个类型

- 预测：将可能以“对等子云”的形式出现在云计算中。
- 依据：云计算对超大规模、多类型资源的统一管理是困难的；对等计算具有鲁棒性、可扩展性、成本、搜索等方面的优点



Google的云计算服务曾出现严重问题，Gmail、Blogger和Spreadsheet等服务均长时间当机。亚马逊S3云计算服务也曾出现问题。而P2P系统则有更强的抗毁能力。

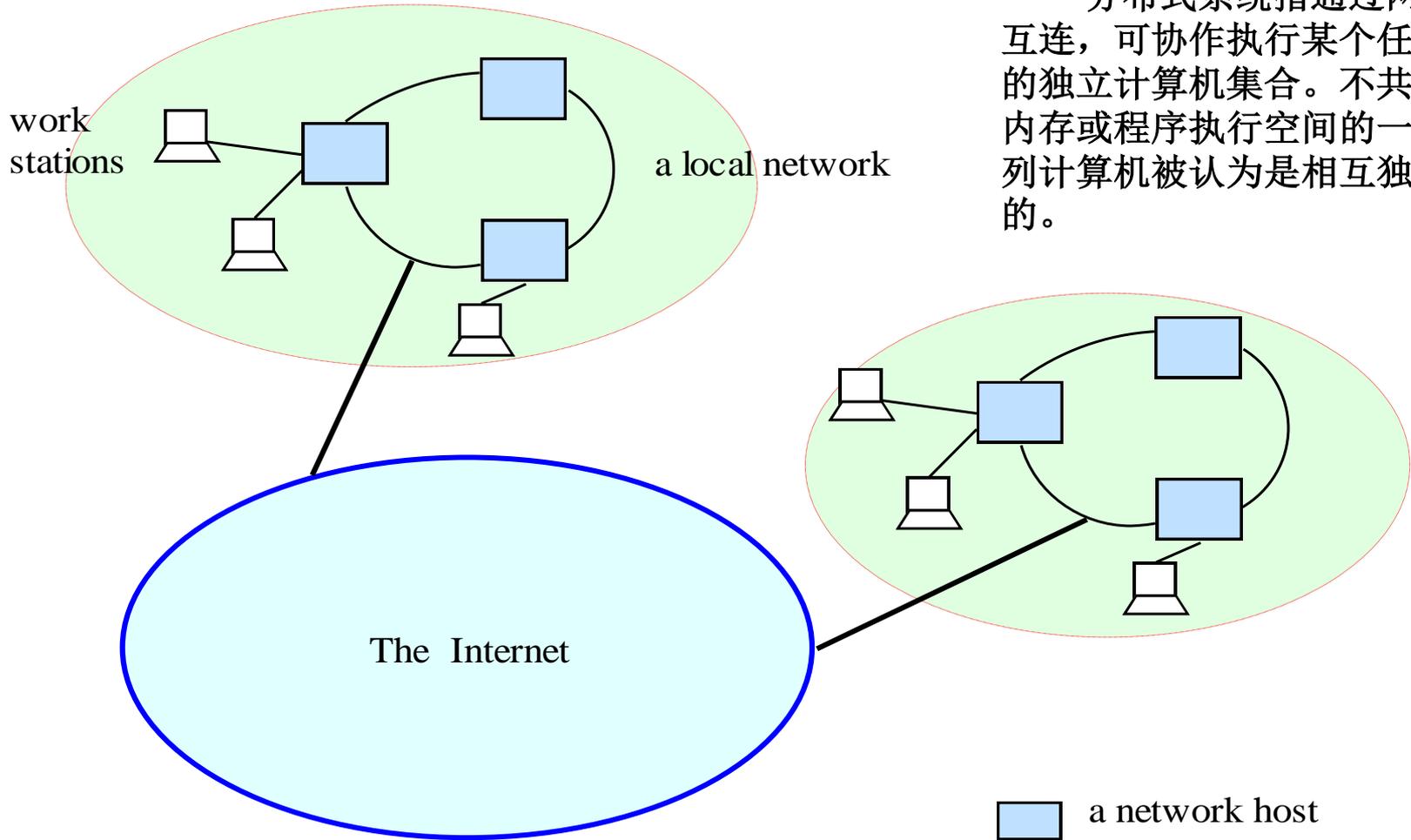
提 纲

- 分布式计算概念
- 分布式系统介绍
- 分布式计算基础技术



分布式系统介绍

分布式系统指通过网络互连，可协作执行某个任务的独立计算机集合。不共享内存或程序执行空间的一系列计算机被认为是相互独立的。





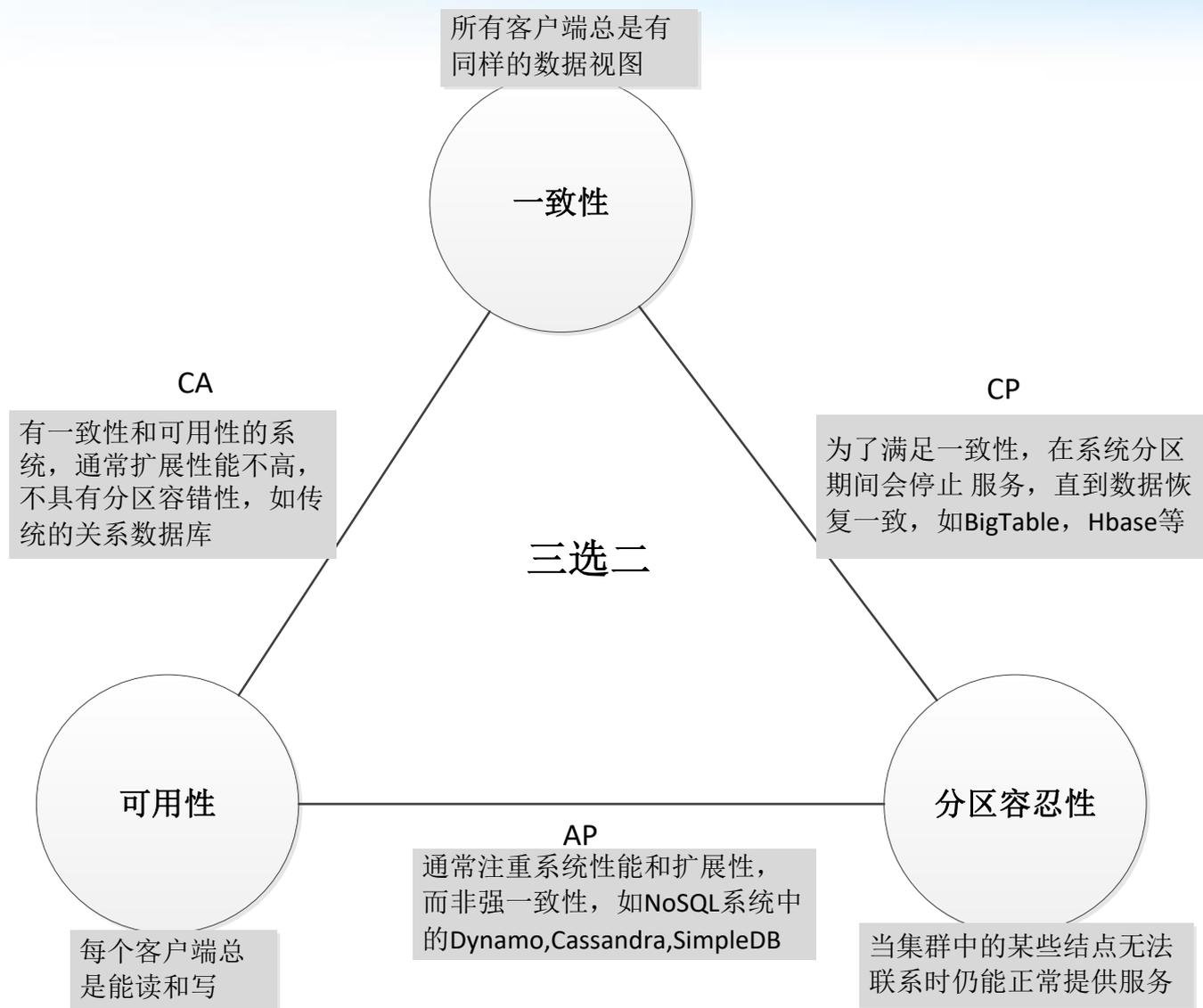
目前因特网上参加人数最多的分布式计算项目**SETI@home**

[著名分布式计算项目介绍](#)

分布式系统特征

- ❖ **可靠性**：指一个分布式系统在它的某一个或多个硬件的软件组件造成故障时，仍能提供服务的能力。
- ❖ **可扩展性**：指一个系统为了支持持续增长的任务数量可以不断扩展的能力。
- ❖ **可用性**：指一个系统尽可能地限制系统因故障而暂停的能力。
- ❖ **高效性**：指一个分布式系统通过分散的计算资源来实现任务执行的高效率。

CAP理论

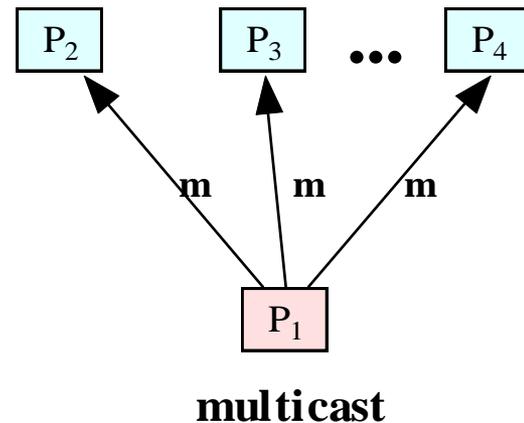
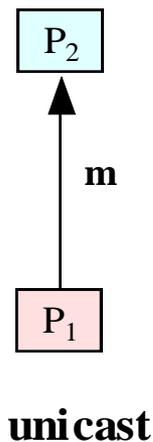
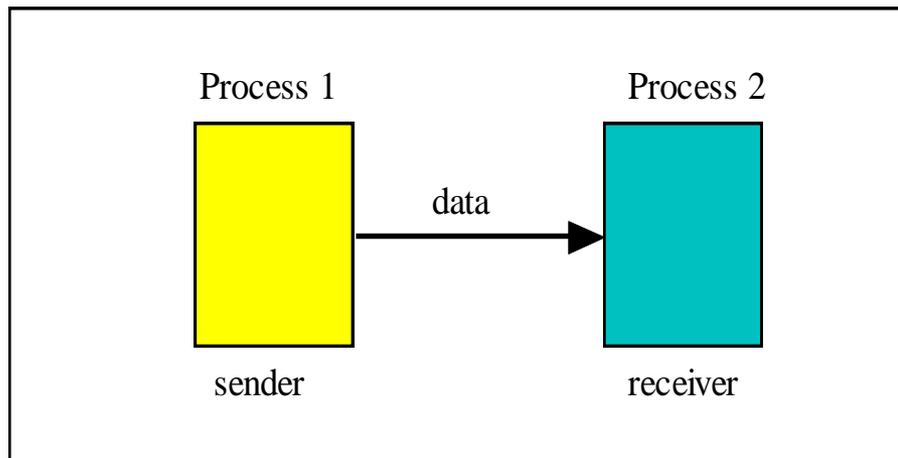


提 纲

- 分布式计算概念
- 分布式系统介绍
- 分布式计算基础技术



进程间通信

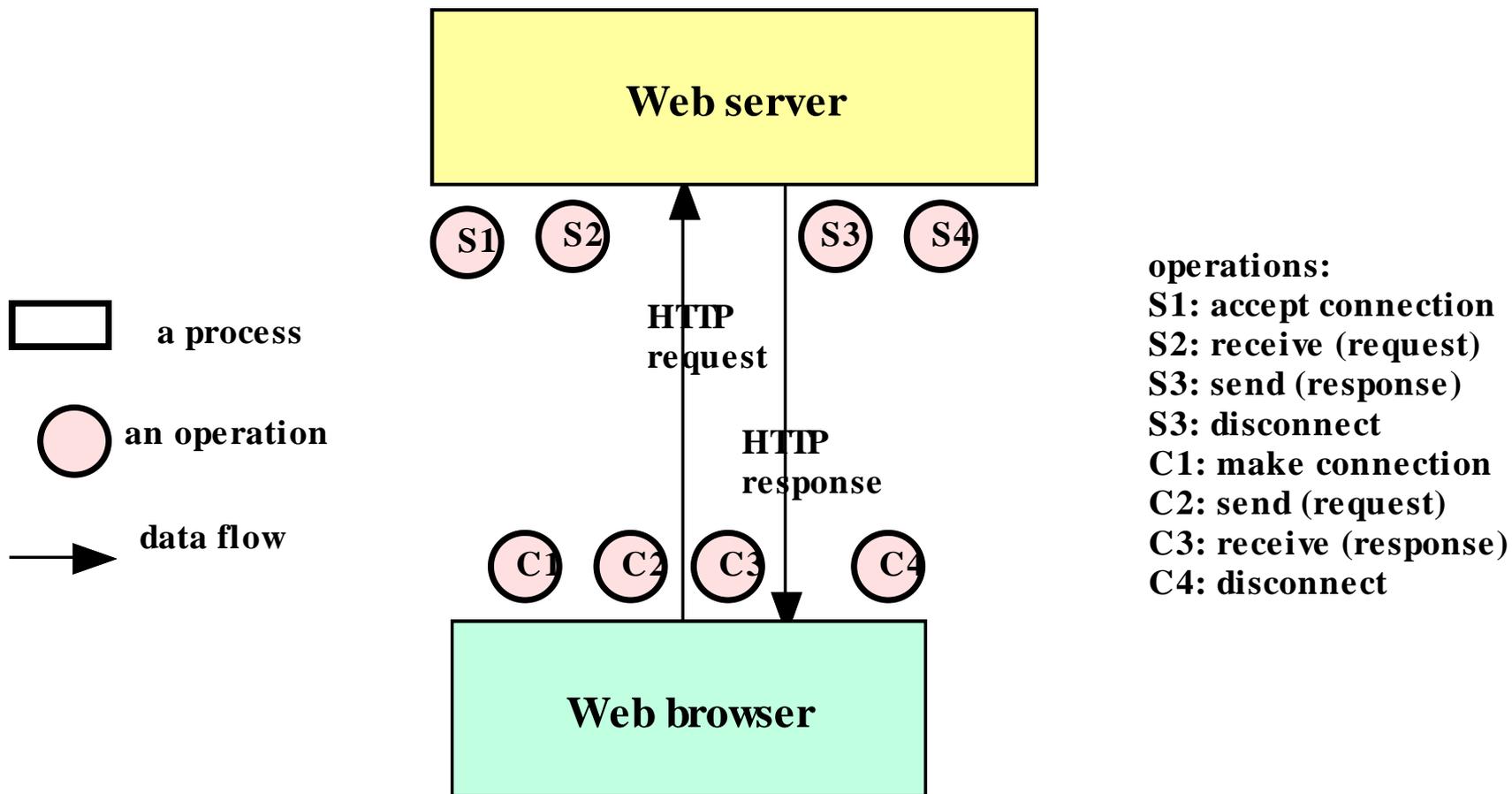


- ◆ 分布式计算的核心技术是进程间通信（**interprocess communication, IPC**），即在互相独立的进程（进程是程序的运行时表示）间通信及共同协作以完成某项任务的能力。
- ◆ 在分布式计算中，两个或多个进程按约定的某种协议进行**IPC**，此处协议是指数据通信各参与进程必须遵守的一组规则。在协议中，一个进程有些时候可能是发送者，在其他时候则可能是接收者。当一个进程与另一个进程进行通信时，**IPC**被称为单播（**unicast**）；当一个进程与另外一组进程进行通信时，**IPC**被称为组播（**multicast**）。

IPC程序接口的四种基本操作

- ◆ 发送（**Send**）。该操作由发送进程发起，旨在向接收进程传输数据。操作必须允许发送进程识别接收进程和定义待传数据。
- ◆ 接收（**Receive**）。该操作由接收进程发起，旨在接收发送进程发来的数据操作必须允许接收进程识别发送进程和定义保存数据的内存空间，该内存随后被接收者访问。
- ◆ 连接（**Connect**）。对面向连接的IPC，必须有允许在发起进程和指定进程间建立逻辑连击的操作：其中以进程发出请求连接操作而另一进程发出接受连接操作。
- ◆ 断开连接（**Disconnect**）。对面向连接的IPC，该操作允许通信的双方关闭先前建立起来的某一逻辑连接。

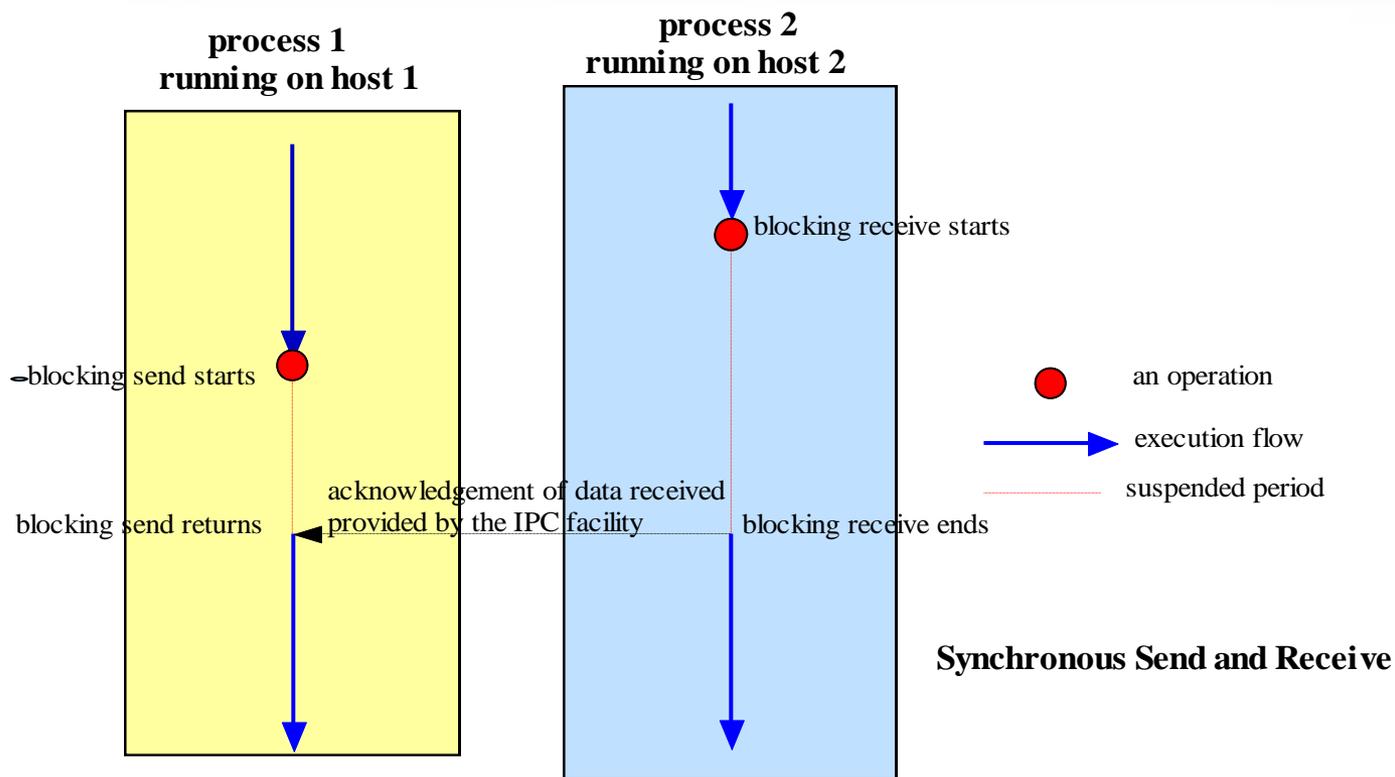
HTTP进程间通信实例



事件同步

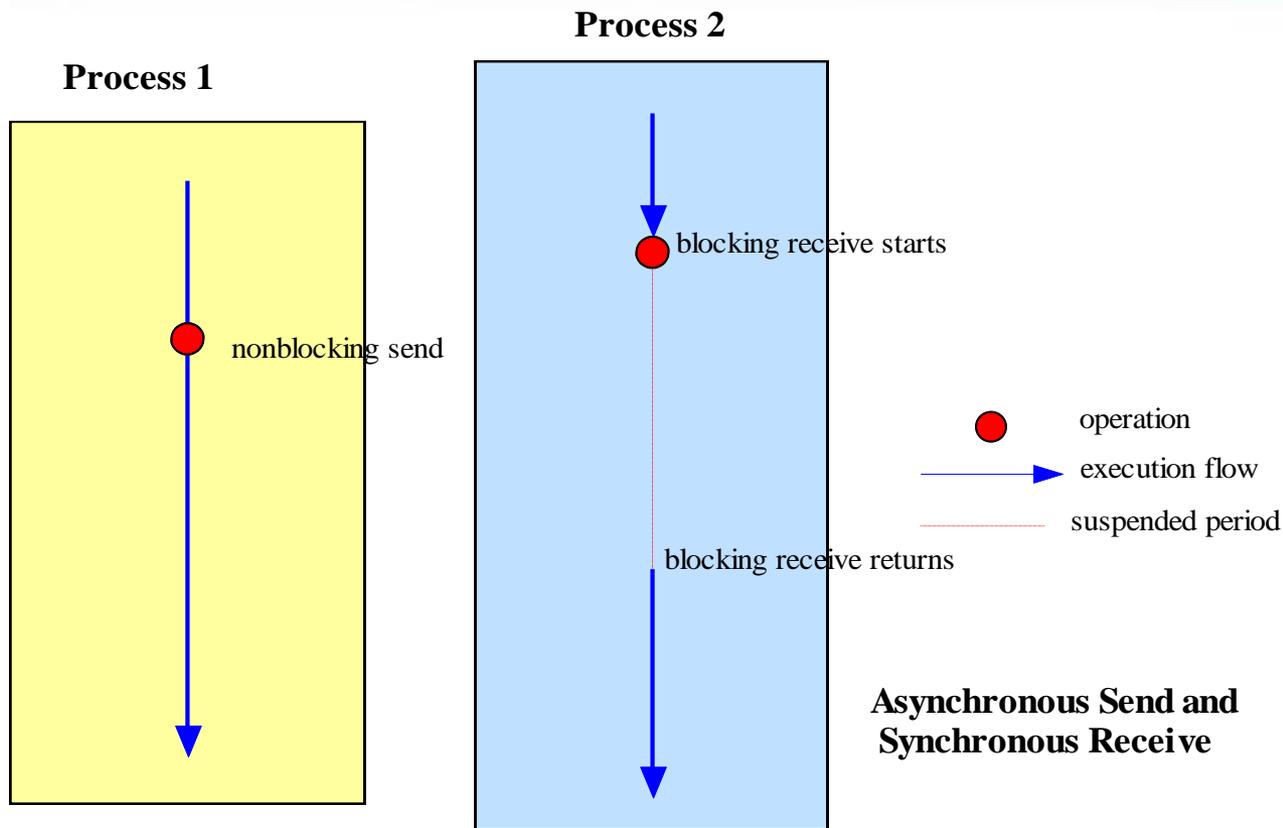
- ◆ **IPC**中的一个主要难点是进行**IPC**的各相关进程是独立执行的，各进程不知道对方进程的情况。协议涉及的双方必须按特定顺序发起**IPC**操作，否则可能通信失败。
- ◆ 因此，参与通信的两个进程需要同步他们的操作，由一方发送数据，另一方则需要等待所有数据发送完成时，开始接收数据。
- ◆ **IPC**设施提供事件同步的最简单的方法是使用阻塞（**blocking**）机制或同步（**synchronous**），即挂起某一进程的执行，直到该进程发起的某个操作执行结束。
- ◆ 另外，**IPC**操作可以是异步（**asynchronous**）或非阻塞操作（**nonblocking**）。进程发起的异步操作不会引起阻塞。因此，一旦向**IPC**设施发出异步操作后，进程可以继续执行。当该异步操作完成后，进程才会随后得到**IPC**设施的通知。

同步send和同步receive



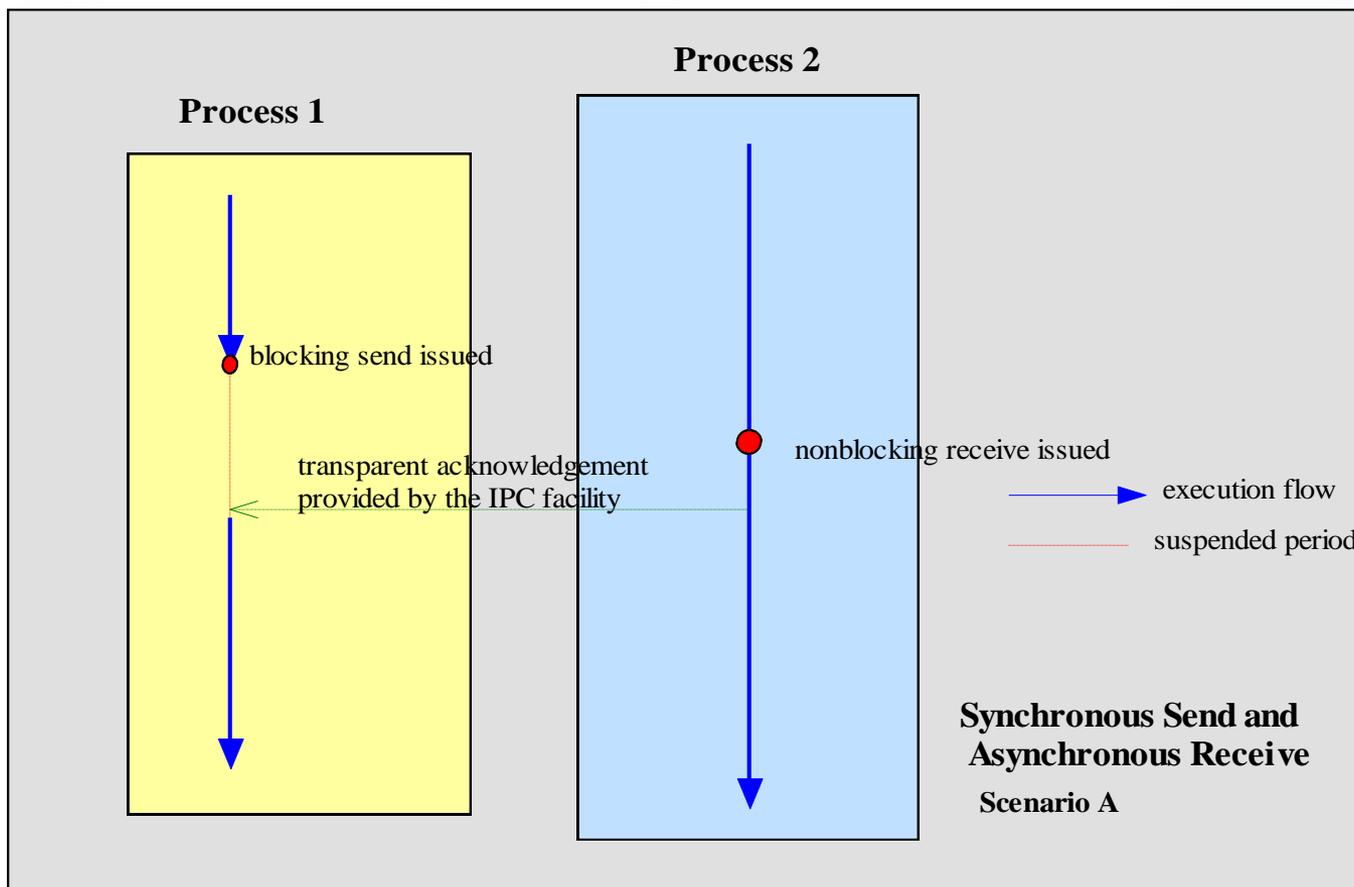
receive操作的发出导致该发起进程挂起，直到接收完成该操作的所有数据。同样地，send操作的发出导致发送进程挂起。当发送的数据被进程2接收后，主机2的IPC设施向主机1的IPC设施发送一条确认信息，进程1随后可被解锁。注意，消息确认由两台主机的IPC设施处理，并且对两个进程是透明的。

异步send和同步receive



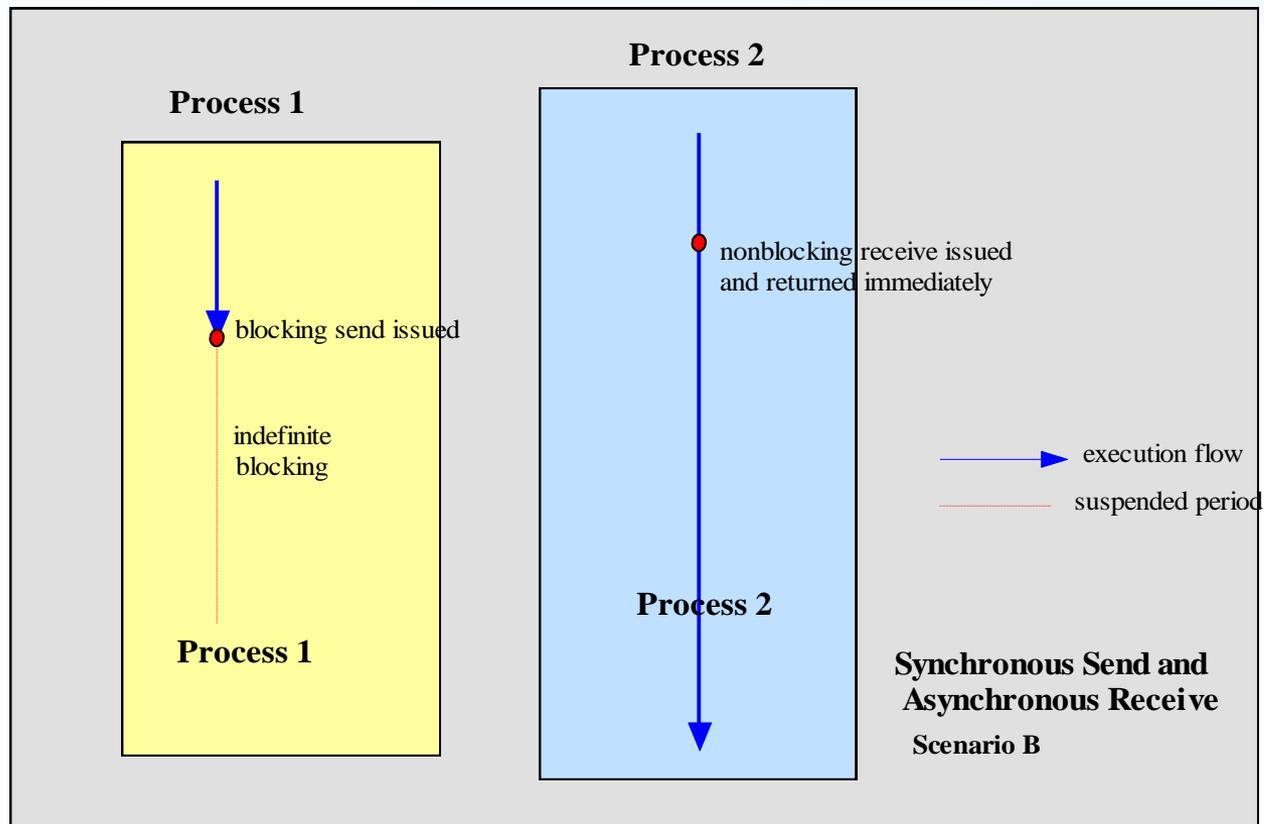
receive操作的发出将导致接收进程挂起，直到接收到满足操作的所有数据为止。然而，send操作的发出不会导致发送进程挂起。在本例中，发送进程永远不会被阻塞，因此，进程2所在主机的IPC设施不必发送确认消息。

同步send和异步receive情形1



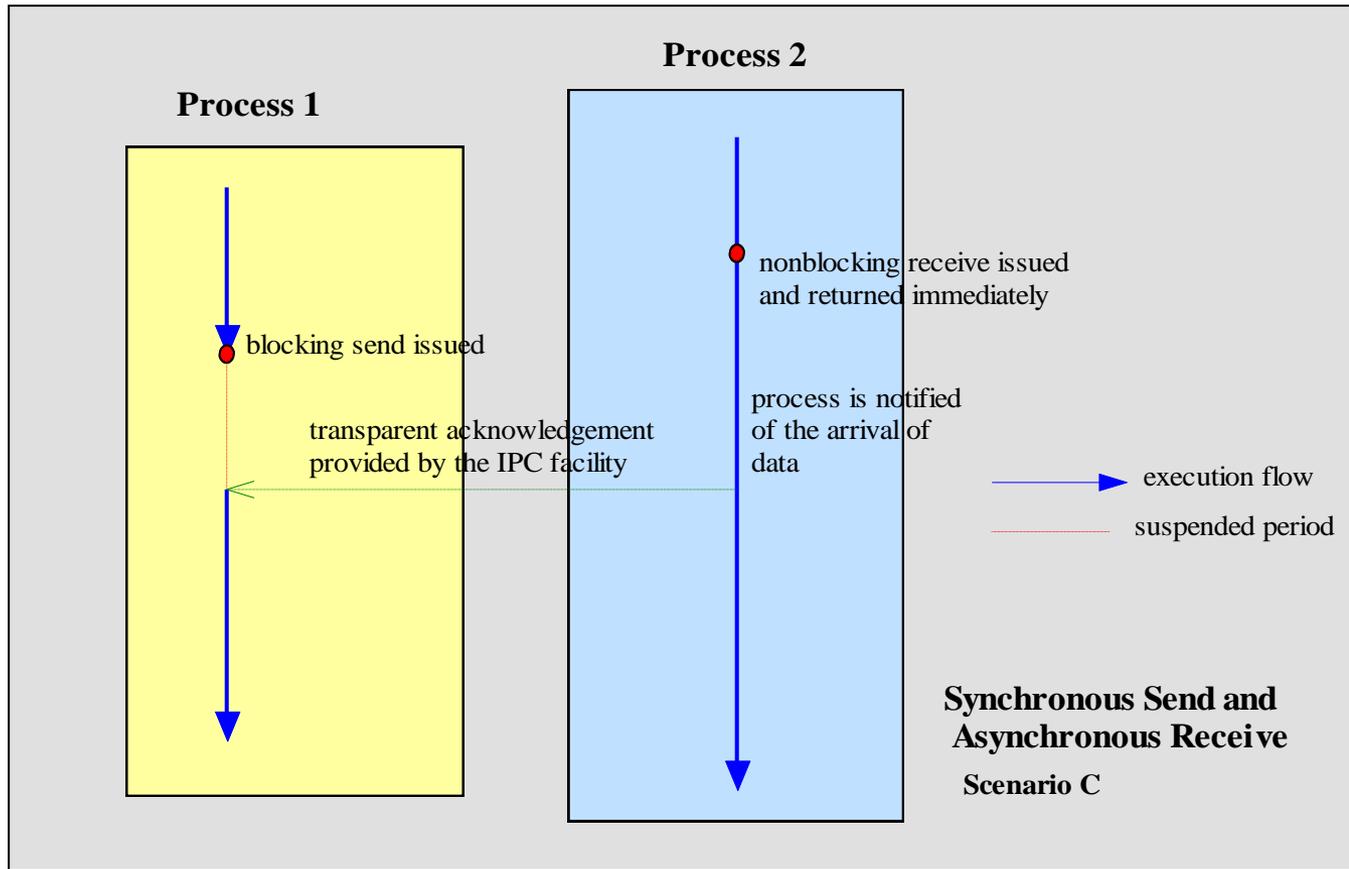
接收操作请求的数据在receive操作发出时已经到达，在这种情况下，数据被立即传送到进程2，主机2的IPC设施返回的确认消息将进程1解锁。

同步send和异步receive情形2



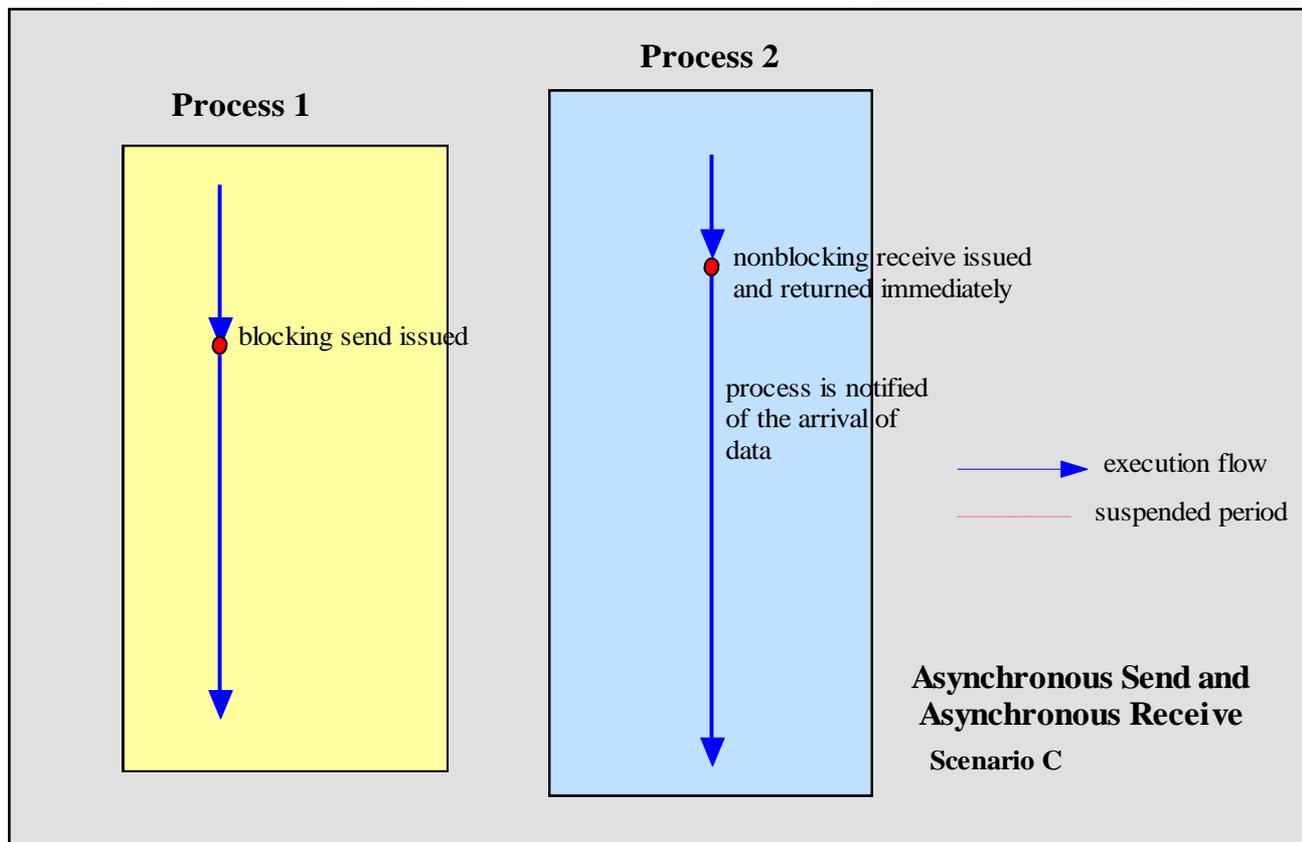
receive操作请求的数据仍未到达；没有数据传递到该进程。接收进程负责确定已真正接收到数据，如果需要的话，重复receive操作，直到数据到达（注意，通常由程序使用循环来重复发出receive操作，直到等待的数据全部接收。这种重复尝试技术被称为轮询技术）。进程1被无限期阻塞，直到进程2重发receive请求，并最终收到主机2 IPC设施的确认消息

同步send和异步receive情形3



receive操作请求的数据仍未到达。当请求数据到达时，主机2的IPC设施将通告进程2，此时进程2可以继续处理数据。该情形要求进程2提供一个可以被IPC设施调用的侦听接口或事件号，用于向进程通告请求数据的到达

异步send和异步receive

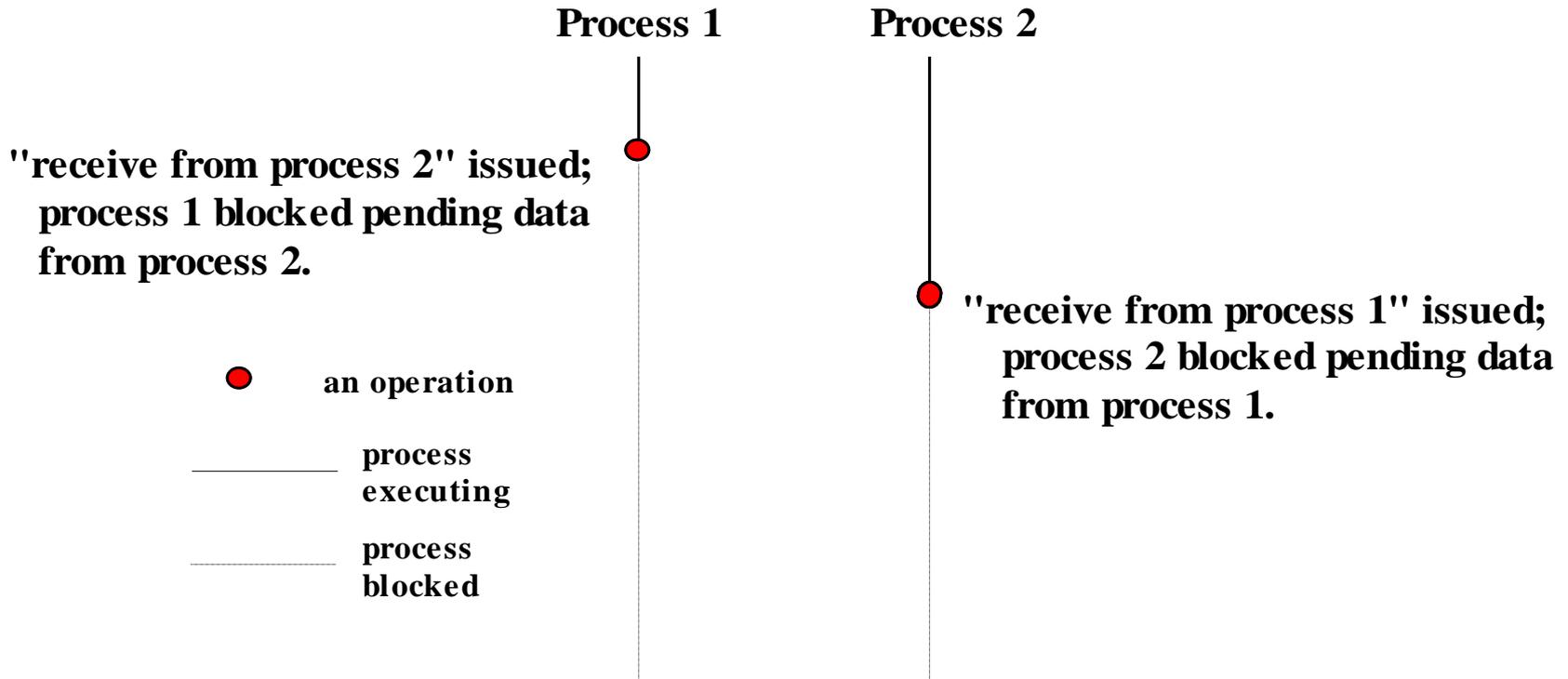


如果双方都没有阻塞，数据能传送到接收者的唯一途径就是由IPC设施保留接收到的数据。接收进程随后可以被通告数据到达了。另外，接收进程也可以轮询数据是否已经到达，并在所等待的数据到达时，对其进行处理

死锁和超时

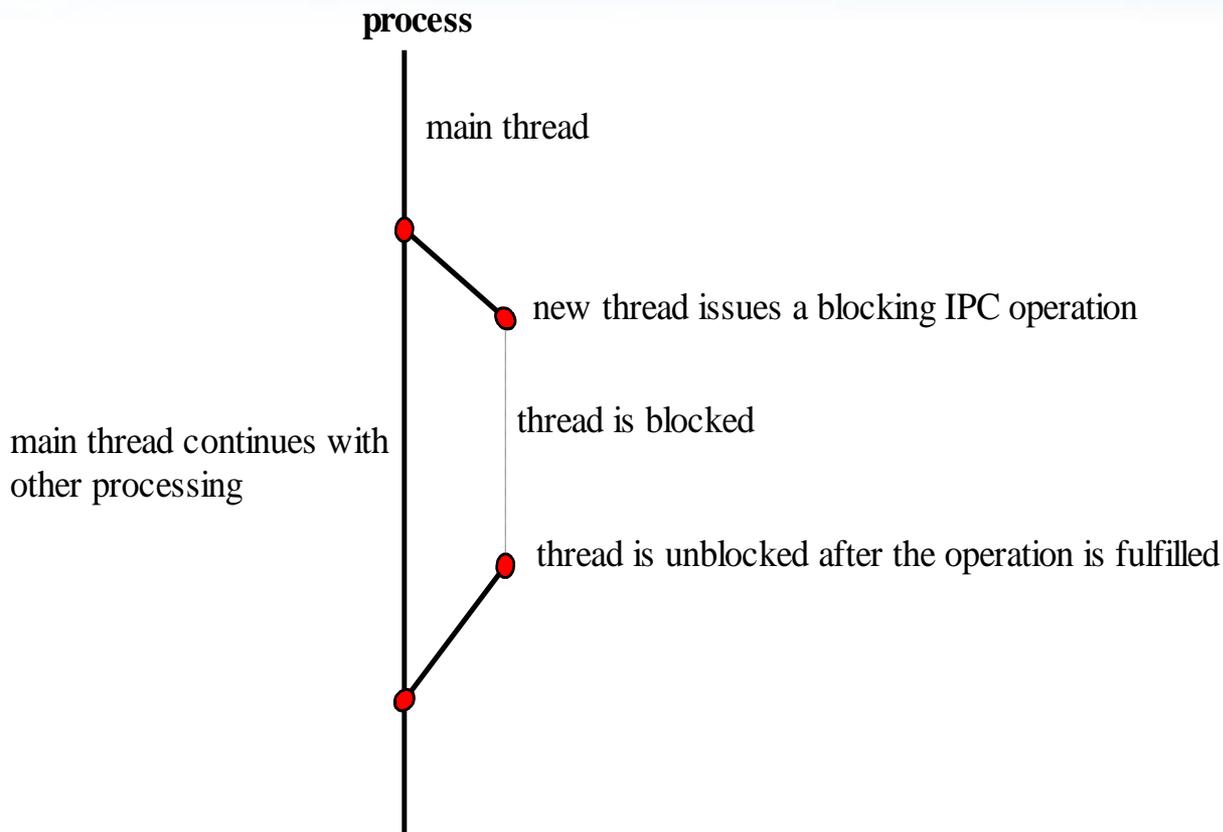
- ◆ 虽然阻塞提供IPC必要的同步，阻塞以错误的顺序发起是可能会引起死锁（**deadlocks**）。
- ◆ 连接和接收操作可能会导致无限期的阻塞（**indefinite blocking**），死锁也可能造成无限期的阻塞。
- ◆ 例如，进程Process1和进程Process2因相互等待对方而无限期阻塞，此时，即由无限期阻塞而产生了进程的死锁。见后页的图示说明。
- ◆ 我们一般不期望被请求的进程被无期限挂起（即无期限阻塞），无期限阻塞可以通过使用超时或子线程来避免。

死锁



尽管阻塞机制为IPC提供了必要的同步，但是同步操作如果按照错误的顺序执行就可能会产生死锁，造成进程被无限期挂起。

用线程实现异步操作



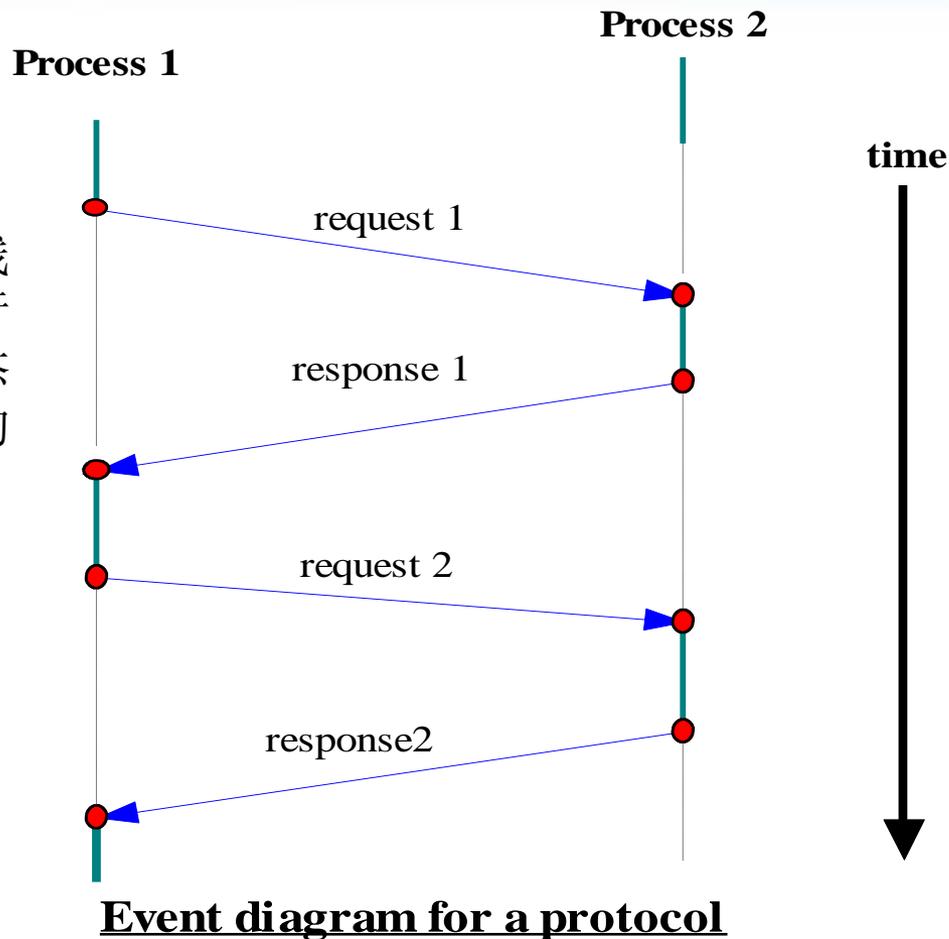
在使用IPC编程接口时，了解该操作是同步操作还是异步操作是非常重要的。如果阻塞操作仅仅是发送或接收，那么编程人员就可以使用子进程或线程来提供阻塞，让程序的主线程或父进程继续执行其他的任务，而子进程或线程将被挂起，直到接收到响应为止，这就是所说的异步处理。



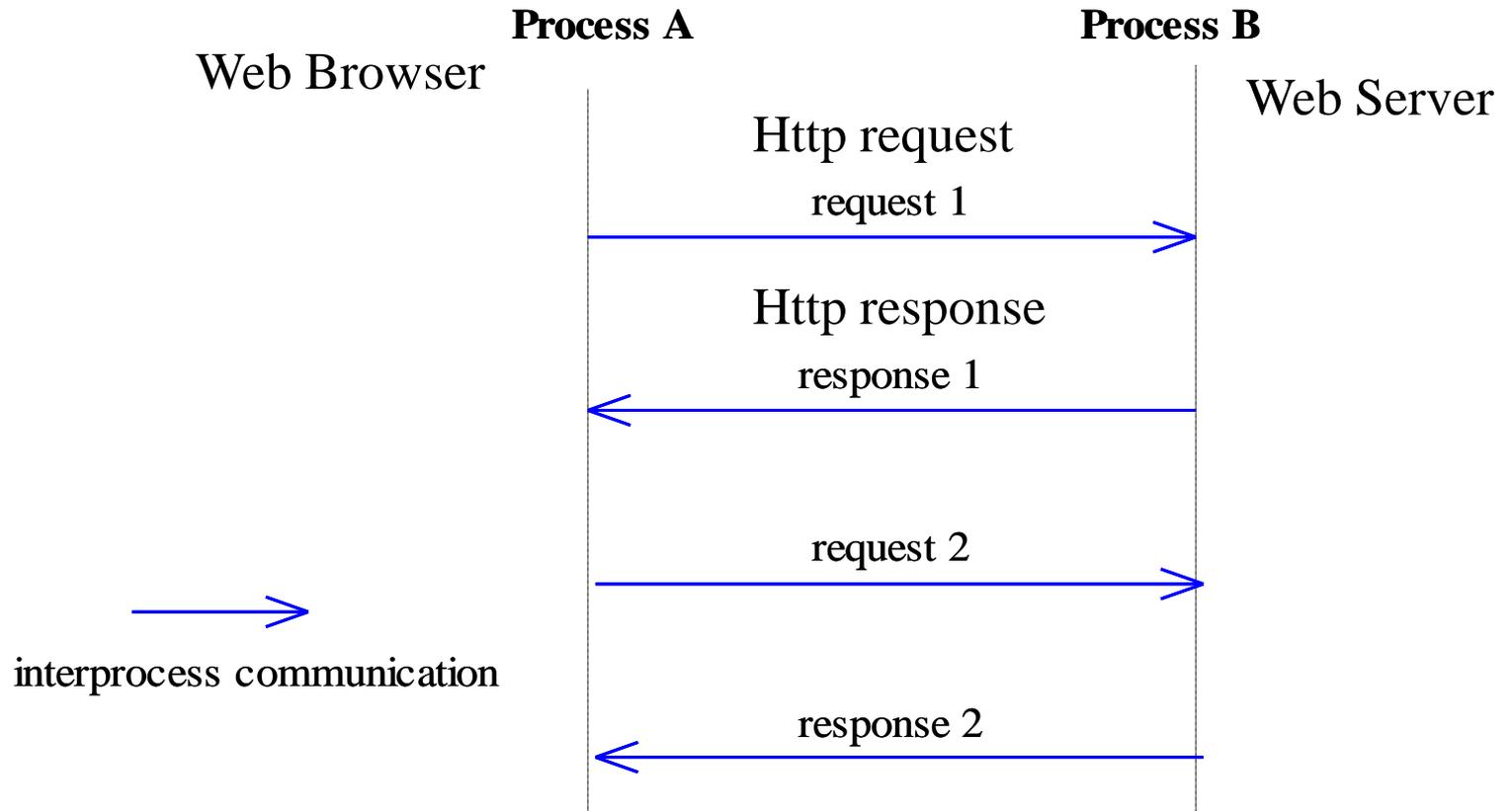
事件状态图

一个包括两个并发进程A和B的请求-响应协议的事件状态图。每个进程随事件变化的执行情况用垂直线表示，时间沿垂直线向下增加。执行线上的实线段表示进程处于活动状态的时间段，虚线段表示进程被阻塞的时间段。

-  interprocess communication
-  execution flow
-  process blocked



HTTP顺序状态图



HTTP会话期间的对话

Script started on Tue Oct 10 21:49:28 2000

9:49pm telnet www.csc.calpoly.edu 80

Trying 129.65.241.20...

Connected to tiedye2-srv.csc.calpoly.edu.

Escape character is '^'].

GET /~mliu/ HTTP/1.0

HTTP Request

HTTP/1.1 200 OK

HTTP response status line

Date: Wed, 11 Oct 2000 04:51:18 GMT

HTTP response header

Server: Apache/1.3.9 (Unix) ApacheJServ/1.0

Last-Modified: Tue, 10 Oct 2000 16:51:54 GMT

ETag: "1dd1e-e27-39e3492a"

Accept-Ranges: bytes

Content-Length: 3623

Connection: close

Content-Type: text/html

<HTML>

document content

<HEAD>

<TITLE> Mei-Ling L. Liu's Home Page

</TITLE>

</HEAD>

<BODY bgcolor=#ffffff>

...

IPC范型

level of
abstraction

IPC paradigms

Example IPC Implementations

remote procedure/method
socket API
data transmission

Remote Procedure Call (RPC), Java RMI

Unix socket API, Winsock

serial/parallel communication

在最低抽象层，IPC利用底层的串行或并行数据传输机制，在连接上传输二进制流。例如，这种IPC范型可以用于编写网络驱动软件。这种形式的IPC属于网络或操作系统编程领域。

下一个抽象层是众所周知的一种范型，称作socket应用程序接口（socket API）。在socket范型中，两个进程使用名为socket的逻辑构造交换数据，每一方都要建立一个socket。待发送数据被写入socket。在另一端，接收进程从自身的socket中读取或提取数据。

远程过程调用（remote procedure call）或远程方法调用（remote method invocation）范型通过允许向远程进程发送过程调用或方法调用，来提供更高层次的抽象。这是，数据作为参数和返回值，在两个进程间进行传递。