基于纠删码的 HDFS 存储方案①

卞艺杰, 马瑞敏, 李亚冰, 吴 慧

(河海大学 商学院, 南京 211100)

摘 要: HDFS 文件系统通过多副本备份的方式解决数据损坏或丢失的问题, 但是随着存储系统内容增多, 在数据量级很大的时候, 这种容灾方案消耗的额外存储空间是实际存储内容的数倍, 不利于系统资源长期积累. 文章提出使用纠删码编/解码文件代替 HDFS 的副本备份容灾策略, 在保证数据安全性的前提下大大提高了存储空间利用率, 降低存储额外消耗.

关键词: HDFS; 纠删码; 副本备份容灾

HDFS Storage Solutions Based on Erasure Codes

BIAN Yi-Jie, MA Rui-Min, LI Ya-Bing, WU Hui

(Business School, Hohai University, Nanjing 211100, China)

Abstract: Through the multiple-backup strategy HDFS can restore data easily when data is damaged or missed. However, the data stored in system increases all the time. When the data scale has become very big, the strategy will need several times of storage space to store the backup data. This article proposes to use erasure codes to replace the multiple-backup strategy, which can greatly improve the storage efficiency and reduce extra storage expend.

Key words: HDFS; erasure code; multiple-backup

分布式服务器集群提供了强大的存储能力和极高的扩展性, 可以通过添加廉价计算机来扩展存储与计算能力. Hadoop 分布式框架是目前开发最成熟、应用最广泛的开源分布式框架, 它的子项目 HDFS 分布式文件系统能够为上层应用提供高吞吐量、可扩展的大文件存储服务. HDFS 文件系统通过多副本备份的方式来避免数据损坏、丢失等情况. 通过这种简单的容错方式, 可以保证当某一节点上的数据失效时, 通过删除、复制的方式就可恢复. 这种方式优点在于实现简单, 安全性高, 但是在数据量级很大的情况下, 就造成了大量空间资源浪费, 需要巨大的额外存储空间消耗. Facebook 经营着目前全球最大的 Hadoop 集群, 2008 年 Facebook 就存储着 100 亿张照片约 1PB 的数据, 到 2013 年 2 月 Facebook 上数码照片的数量已经达到了 2400 亿张, 而且这个数量还在以每天 3500 万张的速度增长[1]. 目前 Facebook 大约存储了 24PB 的照片, 使用 HDFS 的容错方式就需要额外 48PB 的存储空间, 存储消耗相当大, 当数据量级不断增加时, 这种单纯依靠多副本备份的方式显然不能长久地延续. 本文提出使用纠删码编/解码文件代替 HDFS 的副本备份容灾策略, 通过将文件分割编码分布存储在不同节点机器上, 在需要时只需要获得一定量的数据片段就可以恢复整个文件, 在保证数据安全的情况下大大减少冗余数据, 降低存储消耗, 提高了存储系统的空间利用率, 有利于存储系统长期的资源存储和积累.

1 HDFS容错管理研究

1.1 HDFS 多副本容错机制

在数据存储的过程中出现丢失或损坏是不被允许的而且致命的, HDFS 在设计之初就默认集群中节点失效属于常态, 因此设计了比较完善的多副本容错管理机制, 具体的容错机制如下[2][3]:

HDFS 中的节点有两类, 一类是管理节点(Namenode), 负责文件系统的命名空间管理, 监听、处

① 收稿时间:2014-03-11;收到修改稿时间:2014-04-08

理请求, 心跳检测和集群配置信息等. 另一类是数据节点(Datanode), 存储数据块. HDFS 将文件分成固定大小的数据块, 对于每个数据块, 系统默认存储 3 个副本. HDFS 首先确定客户端的位置, 如果客户端在集群中, 则将第一个副本保存在客户端所在的节点中; 如果客户端在集群之外, 则选择集群中剩余存储容量较多比较空闲的节点中. 第二个副本与第一个副本分开存储, 一般放置在不同的机架中的某一节点. 第三个副本与第二个副本存放在相同的机架, 但是不同节点中. 集群中 Datanode 会定期地向 Namenode 发送心跳报告, 如果 Namenode 收到了 Datanode 节点定期发送的心跳报告, 就表明该节点工作正常; 否则表明该节点失效, Namenode 将标记此数据节点, 对该数据节点的读取请求将被拒绝. 这时 Namenode 会进行协调, 从保存有失效数据块的数据节点进行复制, 传送到新的数据节点. 这种文件存储策略保证了系统强大的数据恢复能力, 在用户数据丢失或损坏时, 系统会删除失效数据块, 从其他节点完全复制数据块.

1.2 HDFS 容错机制存在的问题

通常的冗余备份机制包括 2 类: 一类是多副本容错机制, 即镜像方法; 而另一类就是纠删码的方法[4]. HDFS 则是采用简单的多副本容错机制来解决数据容错的问题, 但是随着数据量级的不断扩大, 这种容错管理的弊端也逐渐显现出来.

假如备份数为 3, HDFS 中实际需要的存储空间是存储内容本身的 3 倍, 其存储空间利用率 SE=1/3≈33.3%, 显而易见其存储空间利用率很低, 当对数据安全性要求增高时, 只能通过增加副本数的方法来实现, 存储利用率继续降低, 这种策略不适合集群的长久发展. 副本备份的存储策略冗余度大, 空间利用率低, 只能通过增加文件副本数量来提高文件的容错性能, 随着时间的推移, 数据存储量呈指数增长, 完全备份需要消耗大量的存储空间, 容灾的经济性很低.

但是, 另一种冗余机制——纠删码技术则是将文件通过纠删编码分割成为一系列包含冗余信息的数据片段分布存储在集群中的计算机上, 当需要获取文件时, 只要收集 R 一定数量的数据片段就可以完全恢复文件, 纠删码技术只需要少量的额外空间存储冗余信息, 在保证数据安全性的前提下大大提高了存储空间利用率, 降低存储额外消耗.

2 纠删码与副本容灾的比较分析

2.1 纠删码相关概念

纠删码(Erasure Codes)是一种前向纠错 (Forward Error Correcting, FEC 简称纠错码) 技术, 纠错码纠正的误码位置一般是未知的, 误码位置可知的错误称为删除错误, 纠正删除错误的纠错码称为纠删码[5][6]. 纠删码的核心思想是将包含 k 个数据块的文件编码成为 n(n>k)个数据块, 编码前后数据块大小相同 , 新的数据块中包含了冗余信息, 将这 n 个数据块进行存储; 需要重新获取文件时只需从这 n 个数据块中收到大于等于 k 个数据块就可以恢复原始数据.

纠删码的优势在于, 它有内置的数据安全性, 少量数据块的泄漏不会导致整个数据集信息的外泄; 纠删码在存储结点失效时具有很好的恢复能力, 只需要编码数据块的一个子集就可以恢复原始数据. 当使用纠删码时, 甚至允许系统同时发生多种故障, 如托管设备、存储元件、HDD 和网络等[7].

2.2 纠删码与副本容灾的性能分析

1)相关参数表示

μ, 分布式集群中每个节点可用概率. 一些故障会造成存储节点失效, 在读取数据时若出现错误便无法获取文件. 为了方便计算, 假设每个节点可用概率相同.

k', 文件进行纠删编码后允许丢失的最大数据块数量.

A, 能够成功获取文件的概率, 简称文件获取概率. 在所有副本都丢失或损坏, 纠删编码数据块丢失超过 k'个的情况下将造成文件的不可恢复导致无法获取, 假设其他情况都可以获取文件.

k, 文件在进行纠删编码前被分成的数据块数.

R, 存储消耗系数, 文件需要的存储空间大小与原始文件大小之比, 对于副本备份等于副本数量, 对于纠删码容灾策略即其编码系数 R=n/k.

n, 文件在进行纠删编码后的文件块数, 也可以直接表示为 R*k.

2)文件获取概率

假设副本备份容灾策略的备份数为 R. 由于所有节点之间是相互独立的, 从任何一个存储有该文件的节点都可以实现文件的完全恢复, 所以可以用如下公式来计算文件获取概率[8]:

$$A_{\text{replication}}(R)=C_R^1\mu^1(1-\mu)^{R-1}+C_R^2\mu^2(1-\mu)^{R-2}$$

$$+\ldots+C_R^{R-1}\mu^{R-1}(1-\mu)\ +C_R^R\mu^R$$

这个公式可以简化表示为:

$$A_{replication}(R)=\sum_{i=1}^{R}C_R^i\mu^i(1-\mu)^{R-i}$$ 公式 2-1

使用纠删码容灾策略, 对于(n , k)纠删码一般规定需要 k 个数据块就可以完全恢复文件, 存储消耗系数为 R, 编码后数据块数为 R*k. 这些数据块是分散存储在各个节点上, 相互各自独立, 互不影响, 所以可以用如下公式来表示其文件获取概率[9]:

$$A_{\text{erasurecode}}(k)=C_{R*k}^{k}\mu^{k}(1-\mu)^{R*k-k}+C_{R*k}^{k+1}\mu^{k+1}(1-\mu)^{R*k-(k+1)}+\ldots+C_{R*k}^{R*k}\mu^{R*k}$$

可以简化表示为:

$$A_{erasurecode}(k)=\sum_{i=k}^{R*k}C_{R*k}^{i}\mu^{i}(1-\mu)^{R*k-i}$$ 公式 2-2

在 k=1 的情况下, 可以得到, $A_{replication}(R)=A_{\text{erasurecode}}(k)$ 因此可以使用 A 来表示通用的成功获取文件的概率. 假定 HDFS 集群规模足够庞大, 集群中节点的数目远远大于文件编码后的数据块数 R*k, 保证所有数据块分布在不同的节点, 以及不同数据块间的独立性.

3)性能分析

在公式 2-1 与公式 2-2 的基础之上可以对纠删码冗余、副本备份冗余这两种方式进行简单的计算比较.

举例来说, 取 R=2, μ=0.8, 由公式 2-1 可以计算得到副本备份的文件获取概率 A=0.96.

在公式 2-2 中取 k=2, 得到纠删码冗余的文件获取概率 A=0.9728. k 在一定条件下与 A 是正相关关系, k 越大则 A 相应越大, 这在分布式集群中也很容易理解, 分块越多, 则存储到的节点越多, 节点失效的影响也就越不明显.

因此简单地来看使用纠删码冗余具有更高的性能. 在 R 相等即存储消耗相同的情况下, 纠删码冗余的文件获取概率更高.

本文主要关注的是尽量减少因存储冗余带来的额外消耗问题, 存储系统的基础是 HDFS, 这里假定文件获取一定能够成功, 即文件获取概率为 100%, 对公式(1)和(2)进行比较分析,

$$1=\sum_{i=1}^{R}C_R^i\mu^i(1-\mu)^{R-i}\qquad(1)$$

$$1=\sum_{i=k}^{R'*k}C_{R'*k}^{i}\mu^{i}(1-\mu)^{R'*k-i}\qquad(2)$$

μ 的值与集群状况有关而与选择哪种容灾策略无关, 所以其值在这两种情况下都是相同的, 这就很容易得到 R=R'*k, 这里 k>1, 故有 R'<R, 纠删码冗余的存储系数比副本备份的小, 也就是说纠删码需要更少的数据冗余就可以恢复文件. 系统的存储空间利用率分别为 1/R、1/R', 故纠删码冗余具有更高的存储空间利用率, 在这一方面优于副本备份.

HDFS 分布式文件系统为大规模数据提供存储的同时再提供额外数倍的空间去存储冗余信息显然不利于系统的长久发展. 一方面需要存储系统具有很高的可扩展性, 这点 HDFS 集群通过添加集群中的计算机就可以做到, 另一方面在存储能力一定的情况下提高存储空间利用率, 通过计算分析, 纠删码能够在一定程度上解决这个问题, 在文件可获取率一定的前提下存储空间利用率优于副本备份.

3 基于纠删码改进HDFS存储方案

3.1 设计思路

本文提出基于纠删码容灾的改进方案, 对上传的资源有选择地进行纠删编码, 部分或完全替代 HDFS 的多副本备份策略, 能够大大减少因额外存储造成的空间资源浪费.

主要设计思路如下:

1)完善资源上传模块流程设计, 增加文件具体信息的描述, 包括文件安全等级、存储期限、资源类型、用途等信息. 服务器在接收资源存储的请求时, 通过调用资源识别器对文件进行识别, 根据识别得到的结果确定是否需要使用纠删码进行冗余并且返回匹配的纠删码类型.

2)HDFS 具有文件分块机制, 会将上传的文件进行分块, 得到一定数量的 block, 块的默认大小为 64MB, 不足 64MB 的当作一个块来处理.

3)如果确定资源需要使用纠删码冗余, 将 HDFS 文件系统处理后得到的所有 block 传入纠删码的文件分割器, 分割为一定数目的数据片段(fragment, 数量为 k), 其中纠删码(n , k)的数值可以通过配置文件设置. 分割完成之后每个 block 都会得到相同数目(k)的数据片段.

4)根据选择的纠删码算法对 k 个数据片段编码, 得到 n 个包含冗余信息的数据片段. 这些数据片段被保存到不同的 Datanode 上, 并在 Namenode 的命名空间上记录相应的位置信息.

5)进行文件读取时, 通过 Namenode 收集一定数目

k(k<n)的数据片段, 通过纠删码的解码函数恢复所有的数据块(block), 最终获取完整的文件资源.

系统会发生因节点故障等原因造成的数据损坏或丢失等情况, 在收集数据片段的时候如果遇到无效节点, 则跳过此节点到其他节点, 在完成读取后上报丢失数据信息, 对该文件重新进行纠删编码.

3.2 资源识别器

在应用纠删码进行容错管理的存储系统中, 将本地存储资源进行纠删编码运算之前需要进行资源的分类管理, 针对不同的资源采用相应类型的纠删码, 本文实现这个功能的是资源识别器, 识别之后按照一定的标准匹配适当的纠删码算法. 对实时性要求高的资源采用编码速度较快的LDPC码(可根据具体情况选择优化性更高的纠删码), 保证用户能够较为快速地获取资源; 对需要一次存入多次读取的归档类资源, 较多地关注永久存储安全和空间利用率问题, RS 纠删码冗余信息多, 具有较高的数据可靠性, 适合重安全、轻实时性的存储资源.

资源识别器工作流程如图 1, 用户上传资源时, 资源首先在资源识别器中进行处理. 资源识别器在文件元数据信息中添加一个属性 erasure_type, 如果上传文件属于视频、动画、音频、PPT 等有在线播放需求的资源, 其赋值为数值 1, 表示使用编码速度较快的 LDPC 纠删码进行编解码; 如果不是则继续判断文件是否为需要永久保存、安全性要求比较高的文档、书籍、软件等资源, 若判别为是则设置 erasure_type 值为 2, 即使用 RS 纠删码进行编解码; 最后一种情况是为了避免资源识别器中资源类型的设置不够全面等情况, 会出现无法识别类型的资源, 这时需要将 erasure_type 的值设置为 3, 即使用通用类型的纠删码, 由于 RS 纠删码就是一种综合性比较好的纠删码, 也可以将其他情况统一使用 RS 纠删码.

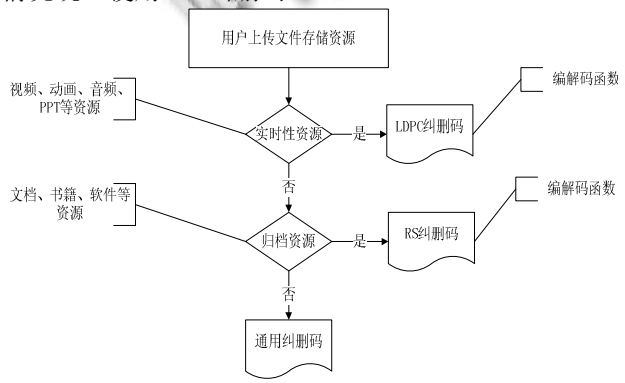

图 1 资源识别器判别流程

3.3 纠删码的数据片段管理

HDFS 中文件被分为一个个数据块(block), 数据块及副本被存储在不同的 Datanode 上, 若使用纠删码容灾, 进行纠删编码之后, 数据块又被分为一个个的数据片段保存在不同的 Datanode 中, HDFS 原来的数据块管理不能够继续适应, 需要进行相应的改进来实现数据片段的有效的管理, 本文通过修改 Namenode 内部 BlocksMap 数据结构来解决上述问题.

Namenode 主要维护着两个映射关系, filename 和 blocksequence(数据块序列)的映射关系、block 与 machinelist(机器列表)的映射关系. 前者通过写入 Fsimage 文件进行保存, 后者主要靠 Datanode 定期发送的报告获知. Fsimage 存储了每个文件对应的所有块的信息, 并没有显示每个数据块保存的 Datanode 列表的信息, 而是通过 Namenode 在一个 BlocksMap 的数据结构中保存数据块与对应 Datanode 列表信息. BlocksMap 的内部数据结构如图 2 所示:

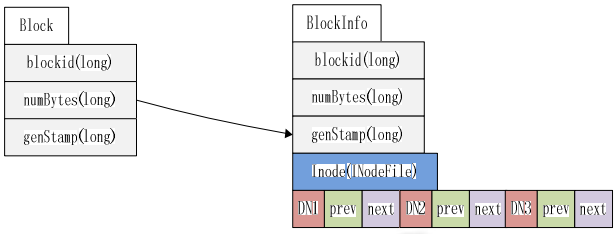

图 2 BlocksMap 内部数据结构

通过上图可以看出, BlocksMap 实际上就是一个 Block 对象对应 BlcokInfo 对象的 Map 表(<Block,BlockInfo>), BlockInfo 是 Block 的一个继承对象, 除了包含Block的所有信息外, 还包含了代表该数据块所属的文件的InodeFile对象引用以及保存有该数据块的 Datanode 列表信息.

图 2 中 DN 代表 Datanode, 因为 HDFS 文件系统默认每个数据块有三个副本, 所以图中共有 DN1、DN2、DN3 三个数据节点. prev 和 next 分别指该数据块在所在数据节点上的前/后一个 BlockInfo 引用. Namenode 采用<DN,PREV,NEXT>三元组结构在内存中保存数据块与存储节点的映射关系, Namenode 中通过上图的方式来保存 block->Datanode list 的对应关系, 当需要查询Datanode->block list的对应关系时, 只需要沿着该数据结构中 prev 和 next 的指向关系就可以得到.

本文在 HDFS 将文件分为数据块的基础上, 对每

个数据块进行纠删编码，把数据块分成一个个数据片段存储在不同的 Datanode 上，这种情况 Fsimage 文件无需作出修改，仍然保存文件与数据块列表的映射关系，但是对每一个数据块对应的 BlockInfo 因为不再使用 Block 进行存储就要做出相应的修改．修改后的 BlockInfo 数据结构如图 3.

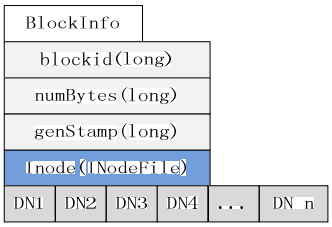

图 3　修改后 BlocksInfo 数据结构

如图所示, BlockInfo 里保存一个 Datanode 的对象列表，保存该数据块编码后生成的所有数据片段的存储节点信息，HDFS 根本无需知道某一数据节点存储了哪些数据片段，因为在实际应用中单一数据片段是毫无意义的，所以删除了 prev 及 next 元素．通过这种方式将所有数据片段的信息保存在 Namenode 的内存中．当需要读取文件时，首先在要在 Fsimage 中查找文件分割后的数据块信息，然后针对每一个数据块(block)，利用 BlocksMap 数据结构内部对应关系<Block, BlockInfo>找到对应的 BlockInfo, BlockInfo 中返回保存该数据块在纠删编码后所有数据片段所存储的数据节点的列表 Datanodes，最后通过依次读取相应的数据节点返回数据片段到解码器完成文件的恢复还原．

3.4 存储层架构

改进后 HDFS 的存储层按照典型的三层结构设计，分为用户接口层、业务逻辑层和数据存储层．用户接口层主要提供用户对资源上传下载进行操作的界面；业务逻辑层是整个文件读写的功能实现层，主要包括资源识别器、编码器和解码器三部分．数据存储层即存储数据的数据节点集群．具体架构如图 4，其中 W 代表文件写入过程, R 代表文件读取过程:

1)文件写入过程

当用户通过界面提交了需要进行存储的文件资源(W1)，资源首先通过 HDFS 被分为 h 个 block，并将文件与文件的元数据分开存储，资源识别器对文件的元数据进行识别判断(W2)，资源识别器根据文件属性类型选择合适的纠删码类型并将文件传入编码器对每一个数据块分别进行纠删编码(W3)，假设使用的是(n, k)纠删码，则每一个 block 就会被分成 n 个含有数据冗余的数据片段，最后按照一定的放置策略或随机将这 n×h 个数据片段放在不同的数据节点 Datanode 中(W6)，并同时与集群 Namenode 通信更新存储位置信息(W5).

2)文件读取过程

在读取文件时，用户通过操作界面提交读取文件的请求(R1)，Web 服务器接收到请求后通过 HDFS Client(R2)进行文件查找拥有该文件片段的 Datanode 信息(R3)，采用最短距离优先的原则在集群中收集相应的数据片段，收集到 k 个时即停止，所有数据片段经过资源识别器识别判断确定需要使用的解码函数(R4)，文件的恢复是在解码器中实现的(R5)，将这 k 个数据片段进行解码运算，恢复数据块，最后将所有的数据块输出给客户端，完成文件的读取(R6).

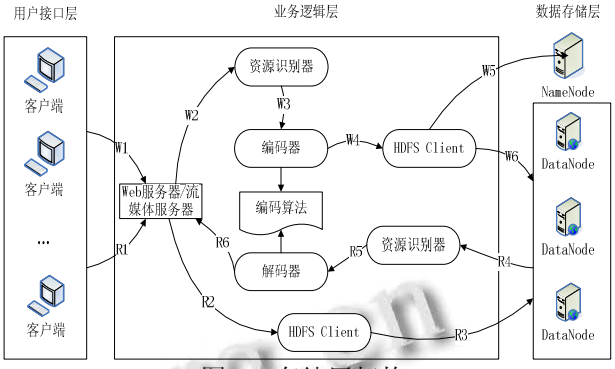

图 4　存储层架构

3.5 实验验证

Hadoop 软件可以作为插件集成到 eclipse 中进行开发应用，HDFS 提供了文件写入和读取 api．本文对 HDFS 读写 api 进行了修改，添加纠删码编解码部分，并修改了数据片段的放置策略．这里使用两台机器上进行模拟实验，分别采用副本容灾和纠删码容灾，副本容灾按照默认的 3 个副本备份的方式，纠删码容灾选用(3, 2)纠删码进行冗余，上传 5 个文件，这里限于集群节点数不足 3 个，令其中一台机器上保存纠删编码后的两个数据片段，存储使用情况如表 1.

表 1　存储使用情况对比

序号	文件大小(MB)	副本备份使用存储(MB)	纠删码冗余使用存储(MB)
1	5	15	7.5
2	20	60	30

3	456.7	1370.1	685.05
4	367.8	1103.4	551.7
5	286.6	859.8	429.9

由表 1 可见使用纠删码作为容灾策略所产生的冗余信息比副本备份要少的多，而且在需要提高数据安全性时副本数更多的情况下差别更为明显.

4 结语

通过分析 HDFS 容错管理机制，发现 HDFS 副本备份容灾策略消耗的额外存储空间是实际存储内容的数倍，不利于系统资源长期积累. 并对纠删码和副本容灾两种策略进行比较分析，提出使用纠删码编/解码文件代替 HDFS 的副本备份容灾策略，基于纠删码改进 HDFS 存储方案，这一方案在保证数据安全性的前提下大大提高了存储空间利用率，降低存储额外消耗.

参考文献

1 Facebook 研发闪存解决照片存出问题. http://tech.sina.com.cn/digi/dc/2013-02-25/09278086522.shtml. 2013.

2 陈宝纯.基于纠删码与 HDFS 的云文件系统[学位论文].吉林大学,2012.

3 HadoopHDFS.http://hadoop.apache.org/docs/r1.0.4/hdfs_ design.html

4 罗象宏,舒继武.存储系统中的纠删码研究综述.计算机研究与发展,2012,49(1):1–11.

5 郭春梅,毕学尧.纠删码的分析与研究.信息安全与技术,2010,(7):38–42.

6 艾达.视频通信抗分组丢失技术研究[学位论文].西安电子科技大学,2006.

7 陈宝纯.基于纠删码与 HDFS 的云文件系统[学位论文].长春:吉林大学,2012.

8 Lin WK, Chiu DM, Lee YB. Erasure code replication revisited. Proc. of the Fourth IEEE International Conference on Peer-to-Peer Computing. 2004.

9 Bhagwan R, Moore D, Savage S, Voelker GM. Replication strategies for highly available peer-to-peer storage. Proc. of FuDiCo: Future directions in Distributed Computing. 2002.