说 明 书 摘 要

本发明实现了一种面向后E级超级计算机的高性能图计算技术，该技术主要包括高能效分布式异步图处理和层次化超大规模分布式通信两个模块。在高能效分布式异步图处理模块，各计算节点将其图分区划分为细粒度的块，并通过核心子图感知的方式优先调度重要的块进行异步图处理，并对加载到缓存的图数据反复处理和利用，从而加快重要图顶点的状态传递，降低数据访问开销和同步开销；在处理每个块时，通过动态挖掘图结构间的依赖关系，来按照图拓扑顺序调度图数据进行异步图处理，使得图顶点状态沿着图拓扑快速传递，加快任务的收敛速度，提高资源有效利用率；并且，在计算过程中，对图顶点状态更新规则化，将图顶点更新的随机访问转化为顺序访问，提高缓存和内存带宽的利用率，同时合并核心图顶点状态数据的存储和访问，提高数据时间/空间局部性，保证每次加载到cache的图顶点状态数据大多都是有效的；当计算节点上的图数据局部收敛并且没有达到通信要求时，生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系，以此来进一步加快图顶点状态的传递速度，提高计算资源的有效并行度。在层次化超大规模分布式通信模块，采用一种基于分组的层次化树型通信模式，并且通过社区结构感知的图分区方式将图数据划分到各计算节点，以此减少通信开销；在通信时，利用逐级层次通信的方式在计算节点间同步图顶点状态数据，并且每个计算节点的通信只有在其图数据以及其管辖计算节点的图数据都计算到局部收敛后才触发，以此减少全局同步开销，充分挖掘各计算节点的计算能力，并有效合并通信消息来降低通信量；同时，发往相同计算节点的消息也被合并在一起统一发送以充分利用网络带宽，减少频繁的小消息通信开销；此外，对通信消息进行压缩，通过去掉图顶点ID等不必要的信息来进一步降低通信量。

权 利 要 求 书

1. 一种面向后E级超级计算机的高性能图计算技术，包括以下步骤：

（1）高能效分布式异步图处理。在处理之前，各计算节点将本地图分区划分为细粒度的块，并根据块中图顶点度数来区分高度数块和低度数块，然后选取高度数块来构建核心子图；同时，识别核心图顶点，并将核心图顶点的状态数据合并存储到一起，提高数据时间/空间局部性，保证每次加载的数据大多都是有效的，减少数据访问开销。在处理时，各计算节点优先选取核心子图中包含活跃图顶点的块进行处理，从而使得重要图顶点的状态快速传递，加快难以收敛的图顶点状态的收敛速度，避免图计算的长尾效应；同时，各计算节点在处理每个块时，采用一种拓扑结构感知的图处理机制，将该块中的图数据按照图拓扑顺序调度到各核上并行处理，从而有效加快图顶点状态传递、避免无效图顶点处理和减少数据访问开销；每个块需要被处理到局部收敛时才会调度下一个块进行处理，使得加载进到cache的图数据被反复处理直到收敛后才被置换出cache，从而可以通过提高数据时间局部性（即提高数据重用率）来避免数据冗余的反复加载，减少数据访问量，减少全局同步开销，使得图顶点状态传递更快，活跃图顶点数量更多，数据并行度更高，最终提高核的利用率；在每个块的处理过程中，将图顶点状态的更新按照图顶点ID顺序进行排序，此后再将排序后的图顶点状态更新按照排好的顺序应用到相应图顶点来更新其状态值，从而将图顶点状态更新的随机访问转化为规则地顺序访问，减少无效数据访问。每个计算节点上的每一轮图计算都需要将其相应的图分区中所有活跃图顶点计算到局部收敛为止，即该计算节点上的图顶点状态在计算过程中不再发生改变为止。计算节点上的图数据计算到局部收敛后，若还没有达到需要通信的要求，则生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系，通过利用这些生成的状态值依赖关系，能够使得通信后更新的图顶点状态值能够快速传递给更多的图顶点，加快图顶点状态的传递速度，提高计算资源的有效并行度。

（2）层次化超大规模分布式通信。各计算节点的通信采用基于分组的层次化树型通信方式。如图1所示，对计算节点进行分组，然后采用树型的方式对每组计算节点进行管理；随后将图数据进行分区，并将图分区分配到各计算节点，在分区时，采用社区结构感知的划分方式，将图顶点之间依赖紧密的子图分配到同一分组中，从而使得子图内的频繁通信发生在同一分组内部，减少全局通信开销；当每个计算节点（例如，Node y，位于第3层）和其管辖的计算节点（即，Node x，Node x+1，Node x+2和Node x+3，位于第4层的一组计算节点）上的图数据均收敛后，该计算节点（即，Node y）才与其所属的计算节点（即，Node 5，位于第2层）和该所属节点所管辖的其余同层次节点（即，Node y+1，Node y+2和Node y+3，位于第3层）进行通信，实现他与同级之间的图顶点状态同步；并以此类推，直到整个集群负责的图顶点状态收敛。在进行通信时，发往同一图顶点的通信消息会被规约合并来减少通信量。同时，发往相同计算节点的消息也被合并在一起统一发送以充分利用网络带宽，减少频繁的小消息通信开销。此外，对通信消息进行压缩，通过去掉图顶点ID等不必要的信息来进一步降低通信量。



图1 基于分组的层次化树型通信方式

1. 如权利要求1所述的方法，其特征在于，所述步骤（1）包括如下子步骤：

（1.1）各计算节点将其本地图分区划分为细粒度的图划分块*P*，图划分块的大小|*P*|由以下公式决定：|*P*| + \*|*V*|\*|| + |*R|* ≤ |*LLC*|，其中|*G*|为图数据的大小，|*V*|为图顶点的数量，||为每个图顶点状态值的大小，|*LLC*|为L3 cache的大小，|*R|*表示预留的空间。|*P*|的值为满足以上公式的最大正整数值。

（1.2）统计各划分块中图顶点的平均度数=，其中是图划分块P的图顶点集合，是图划分块*P*中包含的图顶点数量，是图顶点的度数；随后将根据阈值*Threshold*来区分高度数块和低度数块，*Threshold*是由用户定义的高度数块的比例来决定，例如：现有100个图划分块，用户定义的高度数块的比例为5%，则将图划分块按照降序排序后，*Threshold*为第5个图划分块的；*ADvg*大于*Threshold*的划分块被判定为高度数块，随后将选取的高度数块来构建核心子图。



图2 核心图顶点状态合并

（1.3）核心图顶点状态合并。首先根据用户自定义的核心图顶点比例来判定核心图顶点，例如，如果图中有10000个图顶点，用户自定义的核心图顶点比例为0.5%，则将图顶点按照其度数的降序进行排序后，前50个图顶点为核心图顶点；随后，如图2所示，每个计算节点将其本地的高度数图顶点状态合并存储在其本地的合并状态表中；为了快速访问到本地的核心图顶点状态值，各计算节点构建相应哈希表，其中每个*key-value*对为<*v*，>，为图顶点*v*的状态值在合并状态表中的地址。

（1.4）在每一轮图计算中，各计算节点选取包含活跃图顶点的图划分块进行处理，并且在选取时，优先选择核心子图中的图划分块，选取图划分块后进入步骤（1.5）。若该计算节点的所有图划分块均收敛（即，该计算节点的图分区中没有活跃图顶点），则表示当前轮图计算完成；如果此时达到了通信条件（即，该计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则触发分布式通信模块来同步不同计算节点上的图顶点状态；如果此时还没有达到通信条件，则该计算节点进入步骤（1.10）。

（1.5）在处理每一个选取的图划分块时，计算节点的每个CPU核选取该块中的一个活跃图顶点并压入其私有队列*Q*中（每个CPU核对应一个队列*Q*），以便以该图顶点作为源顶点来按照图拓扑获取图数据（以广度优先的顺序获取），并按照图拓扑顺序进行异步图计算；如果该图划分块中没有活跃图顶点，则表示当前选取的图划分块已经局部收敛，返回步骤（1.4）。

（1.6）每个计算节点的每个CPU核从其对应的队列*Q*中按顺序取出一个图顶点，然后获取该图顶点的邻居图顶点（与一起构成边）以及相应图顶点的状态值。

（1.7）对获取的图数据进行处理。将每条边的源顶点状态值传递到该边的目的顶点来更新目的顶点的状态值。同时，读取图顶点状态值依赖关系表，通过生成的图顶点状态值依赖关系，将通信后的图顶点状态值快速传递给其他图顶点。每次图顶点状态传递会生成一个图顶点状态更新，即<目的图顶点ID，更新状态值>。队列*Q*中所有图顶点的相关图数据被处理后，所生成的图顶点状态更新会生成图顶点状态更新流。

（1.8）对图顶点状态更新流按照其目的图顶点ID进行排序，然后将排序后的图顶点状态更新，顺序地应用到相应图顶点，即将更新状态值和目的图顶点的当前状态值进行计算，得出目的图顶点的最新状态值。对核心图顶点的状态更新，需要更新其在合并状态表的状态值，从而提高核心图顶点状态访问的空间/时间局部性。如果图顶点的状态值被更新，则该图顶点被设置为活跃图顶点，并压入队列*Q*中。

（1.9）判断队列*Q*中是否还存在活跃图顶点。若是，则返回步骤（1.6）；否则，返回步骤（1.5）。



图2 状态值依赖关系计算方式及图顶点状态值依赖关系表

（1.10）生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系。具体来说，由于图顶点状态值之间的依赖关系通常是线性关系，因此，两个不直接相邻的图顶点状态值的依赖关系可以表示为：=*a*\*+*b*，和为两个不直接相邻的图顶点，和为图顶点和的状态值，*a*和*b*为常量。例如，在图2(a)中，需要通信图顶点状态值的图顶点为，对于SSSP算法，与其他不直接相邻图顶点（例如，）之间的状态值依赖关系如图2(a)所示，其中常量*a*和*b*分别为1和9。该依赖关系的计算方式为，进行两次迭代处理，分别得到两组图顶点和的状态值，即<, >和<, >，随后将根据这两组状态值计算出相应的常量*a*和*b*。

（1.11）图顶点状态值依赖关系存储。当两个不直接相邻的图顶点状态值的依赖关系被计算出来后，相应的参数（即，，，*a*和*b*）被存储在一个图顶点状态值依赖关系表中，如图2(b)。当每次通信后，需要通信状态值的图顶点被更新图顶点状态值后，可以利用生成的图顶点状态值依赖关系，将更新后的图顶点状态值快速传递给更多的图顶点，加快图顶点状态的传递速度和传播范围。

（1.12）在生成图顶点状态值依赖关系的过程中，如果某个时刻达到了通信条件（即，当前计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则返回步骤（1.4）。

1. 如权利要求2所述的方法，其特征在于，所述步骤（2）包括如下子步骤：

（2.1）对计算节点进行分组，构建层次化的树型通信架构。如图1的示例中，每个分组包含4个计算节点（除树型结构根节点对应的计算节点外），随后各分组之间构成如图1所示的树型通信架构。各计算节点（例如，图1中的Node y）在通信时，只与自己所管辖的计算节点（树型结构的子节点，图1中的Node x，Node x+1，Node x+2和Node x+3）、自己所属的计算节点（树型结构的父亲节点，图1中的Node 5）和该所属节点所管辖的其余同层次节点（树型结构的兄弟节点，图1中的Node y+1，Node y+2和Node y+3）进行通信。



图3 社区结构检测

（2.2）对图数据进行分区，随后将图分区分配到各计算节点。具体来说，图顶点之间依赖紧密的子图（称作社区）将被分配到同一分组中，甚至同一计算节点，以此来降低全局通信开销。如图3所示，通过社区检测算法（例如，基于标签传播的社区检测算法）来检测出图结构中的社区关系，随后将具有社区相关的图数据分配到同一分组中（例如，将社区结构0分配给图1中的计算节点Node x，Node x+1，Node x+2和Node x+3）。此外，在进行分区时采用笛卡尔顶点切分方法，在该方法中，只有主代理图顶点同时拥有该图顶点的出边和入边，而代理图顶点只有该图顶点的出边或者入边的一类，如图4（a）所示。按照以上方式，图数据将被切分为图分区，然后各个图分区将分配到超级计算机的各个计算节点。



图4 多计算节点间通信以及消息压缩

（2.3）每个计算节点（例如，图1中的Node *y*，位于第3层）在完成其第*Tj*轮图计算后，将需要通信状态值的图顶点的状态值更新生成消息。如图4(a)所示，Node *y*上的图分区收敛后，需要把其镜像代理图顶点（即图顶点，和）的状态值生成通信消息，以便发送到其他计算节点上相应的主代理图顶点（例如，Node *y+1*上的图顶点，和）来同步不同计算节点上相同图顶点的状态值。

（2.4）当通信消息生成以后，如图5(a)所示，对同一图顶点相关的通信消息进行规约，即将同一图顶点的多个通信消息合并为一个通信消息（例如，对于SSSP算法来说，需要计算出这些通信消息中得状态值更新最小值），从而减少通信量。

（2.5）如图5(b)所示，对发往同一计算节点的信息合并到同一队列以便统一发送，从而将不规则的小消息通信转化为规则的大消息通信，提高网络带宽利用率。



图5 图顶点状态规约和消息合并

（2.6）对通信消息进行压缩。如图4(b)所示，将每个队列中的通信消息进行压缩。具体来说，通信消息根据图顶点ID进行排序，随后通过使用位表来标识图顶点状态值所对应的图顶点ID信息，将消息中的图顶点ID信息去掉，从而进一步减少通信量。

（2.7）该计算节点Node *y*所管辖的计算节点（即，图1中的Node *x*，Node *x+1*， Node *x+2*，和Node *x+3*，位于第4层）和其自身的图数据均收敛后，该计算节点Node *y*将规约、合并和压缩后的图顶点状态数据消息发送到其他计算节点进行通信。具体来说，多机间的通信行为主要分为以下三种行为：

①该计算节点Node y和其所管辖的计算节点（即，图1中的Node x，Node x+1， Node x+2，和Node x+3）之间进行相互通信。具体来说，在通信时，Node y所管辖的计算节点先将他们的通信消息发送给Node y，Node y对通信消息按照步骤（2.4）和步骤（2.5）所描述的方法进行规约、合并和压缩后，Node y再将规整后的消息分别发送给其所管辖的计算节点，从而避免各计算节点之间频繁的小消息通信。

②若按以上方式通信后，相应计算节点的图顶点状态值均不改变（即计算节点Node y和其所管辖的计算节点上的图数据整体达到局部收敛），则计算节点Node y将利用归总后的通信消息与其所属的计算节点（即，图1中的Node 5，位于第2层）和该所属节点所管辖的其余同层次节点（即，图1中的Node *y+1*，Node *y+2*，和Node *y+3*，位于第3层）进行通信，从而同步图顶点状态值。

③若该计算节点Node y和其所管辖的计算节点需要与不直接管理或不直接所属的计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）进行通信时，通信消息会根据树型通信结构进行发送和传递，避免全局的频繁小消息传递。如图6所示，相关通信消息会先发送到Node y进行汇集，随后按照树型层次化通信结构，将汇聚后的消息发送到Node 5，随后再发送到Node y+3，Node y+3最后将相关通信消息分别发送到相应计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）。通过这样的方式，使得全局通信消息被充分规约、合并和压缩，使得网络通信行为更规则。

图6 与不直接管理或不直接所属的计算节点通信

（2.8）当所有计算节点的图数据均收敛后，即当所有计算节点与其所属的计算节点和其所属节点所管辖的其余同层次节点进行通信后，图顶点状态值均不再改变，则计算结束。

说 明 书

一种面向后E级超级计算机的高性能图计算技术

**技术领域**

本发明属于大数据处理的图计算技术领域，具体涉及一种面向后E级超级计算机的高性能图计算技术，通过感知图数据之间的依赖、利用图结构的幂律分布特征和探索异步图计算的优势来挖掘后E级超级计算机在支持大规模高性能图计算的能力。

**背景技术**

由于图能够很好地描述大数据时代事物之间的关联关系，在结构和语义上有很强的表达能力。为此，大量大数据分析应用均利用迭代图处理技术来挖掘各自需要的数据信息。例如，通过单源点最短路径算法来找出事物间的最短路径；通过连通分量算法找出数据中的社区关系。与此同时，随着数据的爆发式增长，图数据的规模越来越大，对计算能力的需求也越来越强烈。例如AI芯片电路图的图顶点规模已达万亿级别。在对大规模图数据的科学计算中，往往需要利用超级计算机来快速高效地获得分析结果。

然而，图计算和传统的科学计算有着极大不同，它会更明显地表现出数据依赖复杂，负载不均衡，数据局部性差，不规则数据访问等问题，这使得传统超级计算机的大数据处理系统在支持大规模图计算时面临着极大的挑战。具体而言，首先，现有面向超级计算机的图计算系统在运行时缺乏对图顶点状态间复杂依赖的充分感知，在并行执行时以图顶点或者边为基本处理单位，各图顶点难以将其最新状态快速有效地传递给其邻居，导致大量无效图顶点状态更新、计算效率低下（不但收敛速度慢而且底层硬件资源消耗大，例如数据的冗余加载导致内存带宽等资源的消耗），使得有限的硬件资源成为大规模图计算执行的性能瓶颈，最终导致系统性能低下。其次，在图计算过程中，数据间复杂且不规则的依赖导致大量不规则的数据访问以及极差的数据局部性（比如传统CPU架构每次访问64字节的数据，其中只有4或者8字节的数据是有效的），造成存储资源和内存带宽的有效利用率低。同时，现有方案在计算时，所有图顶点之间状态传递需要沿着它们之间的依赖关系链串行进行，这使得底层平台上的大量并行计算资源利用率低，而上面运行的图计算任务不但收敛速度慢而且需要高额的数据访问开销以反复加载图数据以对它们进行处理。此外，图数据间的依赖也导致不同计算节点之间频繁且不规则的小消息通信，造成大量冗余的通信消息（例如，数据包的包头和图顶点ID信息）和极大的网络通信开销，限制了大规模图计算的在超级计算机上的性能和拓展性。

**发明内容**

针对现有技术的不足，本发明提出了一种面向后E级超级计算机的高性能图计算技术，解决了大规模分布式环境下计算性能低、拓展性差、通信开销高等问题，极大提升了超计算机在支持大规模图计算时的性能。

为实现上述目的，本发明包括以下步骤：

（1）高能效分布式异步图处理。在处理之前，各计算节点将本地图分区划分为细粒度的块，并根据块中图顶点度数来区分高度数块和低度数块，然后选取高度数块来构建核心子图；同时，识别核心图顶点，并将核心图顶点的状态数据合并存储到一起，提高数据时间/空间局部性，保证每次加载的数据大多都是有效的，减少数据访问开销。在处理时，各计算节点优先选取核心子图中包含活跃图顶点的块进行处理，从而使得重要图顶点的状态快速传递，加快难以收敛的图顶点状态的收敛速度，避免图计算的长尾效应；同时，各计算节点在处理每个块时，采用一种拓扑结构感知的图处理机制，将该块中的图数据按照图拓扑顺序调度到各核上并行处理，从而有效加快图顶点状态传递、避免无效图顶点处理和减少数据访问开销；每个块需要被处理到局部收敛时才会调度下一个块进行处理，使得加载进到cache的图数据被反复处理直到收敛后才被置换出cache，从而可以通过提高数据时间局部性（即提高数据重用率）来避免数据冗余的反复加载，减少数据访问量，减少全局同步开销，使得图顶点状态传递更快，活跃图顶点数量更多，数据并行度更高，最终提高核的利用率；在每个块的处理过程中，将图顶点状态的更新按照图顶点ID顺序进行排序，此后再将排序后的图顶点状态更新按照排好的顺序应用到相应图顶点来更新其状态值，从而将图顶点状态更新的随机访问转化为规则地顺序访问，减少无效数据访问。每个计算节点上的每一轮图计算都需要将其相应的图分区中所有活跃图顶点计算到局部收敛为止，即该计算节点上的图顶点状态在计算过程中不再发生改变为止。计算节点上的图数据计算到局部收敛后，若还没有达到需要通信的要求，则生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系，通过利用这些生成的状态值依赖关系，能够使得通信后更新的图顶点状态值能够快速传递给更多的图顶点，加快图顶点状态的传递速度，提高计算资源的有效并行度。

（2）层次化超大规模分布式通信。各计算节点的通信采用基于分组的层次化树型通信方式。如图1所示，对计算节点进行分组，然后采用树型的方式对每组计算节点进行管理；随后将图数据进行分区，并将图分区分配到各计算节点，在分区时，采用社区结构感知的划分方式，将图顶点之间依赖紧密的子图分配到同一分组中，从而使得子图内的频繁通信发生在同一分组内部，减少全局通信开销；当每个计算节点（例如，Node y，位于第3层）和其管辖的计算节点（即，Node x，Node x+1，Node x+2和Node x+3，位于第4层的一组计算节点）上的图数据均收敛后，该计算节点（即，Node y）才与其所属的计算节点（即，Node 5，位于第2层）和该所属节点所管辖的其余同层次节点（即，Node y+1，Node y+2和Node y+3，位于第3层）进行通信，实现他与同级之间的图顶点状态同步；并以此类推，直到整个集群负责的图顶点状态收敛。在进行通信时，发往同一图顶点的通信消息会被规约合并来减少通信量。同时，发往相同计算节点的消息也被合并在一起统一发送以充分利用网络带宽，减少频繁的小消息通信开销。此外，对通信消息进行压缩，通过去掉图顶点ID等不必要的信息来进一步降低通信量。

上述大规模高性能分布式图计算系统，执行步骤如下：

（1）高能效分布式异步图处理机制。其主要包括如下子步骤：

（1.1）各计算节点将其本地图分区划分为细粒度的图划分块*P*，图划分块的大小|*P*|由以下公式决定：|*P*| + \*|*V*|\*|| + |*R|* ≤ |*LLC*|，其中|*G*|为图数据的大小，|*V*|为图顶点的数量，||为每个图顶点状态值的大小，|*LLC*|为L3 cache的大小，|*R|*表示预留的空间。|*P*|的值为满足以上公式的最大正整数值。

（1.2）统计各划分块中图顶点的平均度数=，其中是图划分块P的图顶点集合，是图划分块*P*中包含的图顶点数量，是图顶点的度数；随后将根据阈值*Threshold*来区分高度数块和低度数块，*Threshold*是由用户定义的高度数块的比例来决定，例如：现有100个图划分块，用户定义的高度数块的比例为5%，则将图划分块按照降序排序后，*Threshold*为第5个图划分块的；*ADvg*大于*Threshold*的划分块被判定为高度数块，随后将选取的高度数块来构建核心子图。

（1.3）核心图顶点状态合并。首先根据用户自定义的核心图顶点比例来判定核心图顶点，例如，如果图中有10000个图顶点，用户自定义的核心图顶点比例为0.5%，则将图顶点按照其度数的降序进行排序后，前50个图顶点为核心图顶点；随后，如图2所示，每个计算节点将其本地的高度数图顶点状态合并存储在其本地的合并状态表中；为了快速访问到本地的核心图顶点状态值，各计算节点构建相应哈希表，其中每个*key-value*对为<*v*，>，为图顶点*v*的状态值在合并状态表中的地址。

（1.4）在每一轮图计算中，各计算节点选取包含活跃图顶点的图划分块进行处理，并且在选取时，优先选择核心子图中的图划分块，选取图划分块后进入步骤（1.5）。若该计算节点的所有图划分块均收敛（即，该计算节点的图分区中没有活跃图顶点），则表示当前轮图计算完成；如果此时达到了通信条件（即，该计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则触发分布式通信模块来同步不同计算节点上的图顶点状态；如果此时还没有达到通信条件，则该计算节点进入步骤（1.10）。

（1.5）在处理每一个选取的图划分块时，计算节点的每个CPU核选取该块中的一个活跃图顶点并压入其私有队列*Q*中（每个CPU核对应一个队列*Q*），以便以该图顶点作为源顶点来按照图拓扑获取图数据（以广度优先的顺序获取），并按照图拓扑顺序进行异步图计算；如果该图划分块中没有活跃图顶点，则表示当前选取的图划分块已经局部收敛，返回步骤（1.4）。

（1.6）每个计算节点的每个CPU核从其对应的队列*Q*中按顺序取出一个图顶点，然后获取该图顶点的邻居图顶点（与一起构成边）以及相应图顶点的状态值。

（1.7）对获取的图数据进行处理。将每条边的源顶点状态值传递到该边的目的顶点来更新目的顶点的状态值。同时，读取图顶点状态值依赖关系表，通过生成的图顶点状态值依赖关系，将通信后的图顶点状态值快速传递给其他图顶点。每次图顶点状态传递会生成一个图顶点状态更新，即<目的图顶点ID，更新状态值>。队列*Q*中所有图顶点的相关图数据被处理后，所生成的图顶点状态更新会生成图顶点状态更新流。

（1.8）对图顶点状态更新流按照其目的图顶点ID进行排序，然后将排序后的图顶点状态更新，顺序地应用到相应图顶点，即将更新状态值和目的图顶点的当前状态值进行计算，得出目的图顶点的最新状态值。对核心图顶点的状态更新，需要更新其在合并状态表的状态值，从而提高核心图顶点状态访问的空间/时间局部性。如果图顶点的状态值被更新，则该图顶点被设置为活跃图顶点，并压入队列*Q*中。

（1.9）判断队列*Q*中是否还存在活跃图顶点。若是，则返回步骤（1.6）；否则，返回步骤（1.5）。

（1.10）生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系。具体来说，由于图顶点状态值之间的依赖关系通常是线性关系，因此，两个不直接相邻的图顶点状态值的依赖关系可以表示为：=*a*\*+*b*，和为两个不直接相邻的图顶点，和为图顶点和的状态值，*a*和*b*为常量。例如，在图2(a)中，需要通信图顶点状态值的图顶点为，对于SSSP算法，与其他不直接相邻图顶点（例如，）之间的状态值依赖关系如图2(a)所示，其中常量*a*和*b*分别为1和9。该依赖关系的计算方式为，进行两次迭代处理，分别得到两组图顶点和的状态值，即<, >和<, >，随后将根据这两组状态值计算出相应的常量*a*和*b*。

（1.11）图顶点状态值依赖关系存储。当两个不直接相邻的图顶点状态值的依赖关系被计算出来后，相应的参数（即，，，*a*和*b*）被存储在一个图顶点状态值依赖关系表中，如图2(b)。当每次通信后，需要通信状态值的图顶点被更新图顶点状态值后，可以利用生成的图顶点状态值依赖关系，将更新后的图顶点状态值快速传递给更多的图顶点，加快图顶点状态的传递速度和传播范围。

（1.12）在生成图顶点状态值依赖关系的过程中，如果某个时刻达到了通信条件（即，当前计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则返回步骤（1.4）。

（2）层次化超大规模分布式通信。其主要包括如下子步骤：

（2.1）对计算节点进行分组，构建层次化的树型通信架构。如图1的示例中，每个分组包含4个计算节点（除树型结构根节点对应的计算节点外），随后各分组之间构成如图1所示的树型通信架构。各计算节点（例如，图1中的Node y）在通信时，只与自己所管辖的计算节点（树型结构的子节点，图1中的Node x，Node x+1，Node x+2和Node x+3）、自己所属的计算节点（树型结构的父亲节点，图1中的Node 5）和该所属节点所管辖的其余同层次节点（树型结构的兄弟节点，图1中的Node y+1，Node y+2和Node y+3）进行通信。

（2.2）对图数据进行分区，随后将图分区分配到各计算节点。具体来说，图顶点之间依赖紧密的子图（称作社区）将被分配到同一分组中，甚至同一计算节点，以此来降低全局通信开销。如图3所示，通过社区检测算法（例如，基于标签传播的社区检测算法）来检测出图结构中的社区关系，随后将具有社区相关的图数据分配到同一分组中（例如，将社区结构0分配给图1中的计算节点Node x，Node x+1，Node x+2和Node x+3）。此外，在进行分区时采用笛卡尔顶点切分方法，在该方法中，只有主代理图顶点同时拥有该图顶点的出边和入边，而代理图顶点只有该图顶点的出边或者入边的一类，如图4（a）所示。按照以上方式，图数据将被切分为图分区，然后各个图分区将分配到超级计算机的各个计算节点。

（2.3）每个计算节点（例如，图1中的Node *y*，位于第3层）在完成其第*Tj*轮图计算后，将需要通信状态值的图顶点的状态值更新生成消息。如图4(a)所示，Node *y*上的图分区收敛后，需要把其镜像代理图顶点（即图顶点，和）的状态值生成通信消息，以便发送到其他计算节点上相应的主代理图顶点（例如，Node *y+1*上的图顶点，和）来同步不同计算节点上相同图顶点的状态值。

（2.4）当通信消息生成以后，如图5(a)所示，对同一图顶点相关的通信消息进行规约，即将同一图顶点的多个通信消息合并为一个通信消息（例如，对于SSSP算法来说，需要计算出这些通信消息中得状态值更新最小值），从而减少通信量。

（2.5）如图5(b)所示，对发往同一计算节点的信息合并到同一队列以便统一发送，从而将不规则的小消息通信转化为规则的大消息通信，提高网络带宽利用率。

（2.6）对通信消息进行压缩。如图4(b)所示，将每个队列中的通信消息进行压缩。具体来说，通信消息根据图顶点ID进行排序，随后通过使用位表来标识图顶点状态值所对应的图顶点ID信息，将消息中的图顶点ID信息去掉，从而进一步减少通信量。

（2.7）该计算节点Node *y*所管辖的计算节点（即，图1中的Node *x*，Node *x+1*， Node *x+2*，和Node *x+3*，位于第4层）和其自身的图数据均收敛后，该计算节点Node *y*将规约、合并和压缩后的图顶点状态数据消息发送到其他计算节点进行通信。具体来说，多机间的通信行为主要分为以下三种行为：

①该计算节点Node y和其所管辖的计算节点（即，图1中的Node x，Node x+1， Node x+2，和Node x+3）之间进行相互通信。具体来说，在通信时，Node y所管辖的计算节点先将他们的通信消息发送给Node y，Node y对通信消息按照步骤（2.4）和步骤（2.5）所描述的方法进行规约、合并和压缩后，Node y再将规整后的消息分别发送给其所管辖的计算节点，从而避免各计算节点之间频繁的小消息通信。

②若按以上方式通信后，相应计算节点的图顶点状态值均不改变（即计算节点Node y和其所管辖的计算节点上的图数据整体达到局部收敛），则计算节点Node y将利用归总后的通信消息与其所属的计算节点（即，图1中的Node 5，位于第2层）和该所属节点所管辖的其余同层次节点（即，图1中的Node *y+1*，Node *y+2*，和Node *y+3*，位于第3层）进行通信，从而同步图顶点状态值。

③若该计算节点Node y和其所管辖的计算节点需要与不直接管理或不直接所属的计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）进行通信时，通信消息会根据树型通信结构进行发送和传递，避免全局的频繁小消息传递。如图6所示，相关通信消息会先发送到Node y进行汇集，随后按照树型层次化通信结构，将汇聚后的消息发送到Node 5，随后再发送到Node y+3，Node y+3最后将相关通信消息分别发送到相应计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）。通过这样的方式，使得全局通信消息被充分规约、合并和压缩，使得网络通信行为更规则。

（2.8）当所有计算节点的图数据均收敛后，即当所有计算节点与其所属的计算节点和其所属节点所管辖的其余同层次节点进行通信后，图顶点状态值均不再改变，则计算结束。

总体而言，通过本发明所构思的以上技术方案与现有技术相比，能够取得下列有益效果：

1. 加快图计算收敛速度：本发明在异步计算过程中，通过优先选择重要图数据块来处理，并且在处理每个块时，按照图顶点之间的依赖关系来加载图数据进行处理，使得重要图顶点的最新状态能够沿着图顶点之间的依赖顺序进行快速传递，加快了重要图顶点的状态传递速度，避免图计算的长尾效应和大量无效图顶点处理，从而充分利用底层硬件的高并行计算资源；其次，各计算节点的每轮计算都将其图分区计算到局部收敛为止，这也加快了每轮图计算中图顶点状态的传播范围，从而通过提高数据时间局部性（即提高数据重用率）来避免数据冗余的反复加载，减少数据访问次数和访问量，并且减少全局同步开销，使得图顶点状态传递更快，活跃图顶点数量更多，数据并行度更高，提高核的利用率；通过生成和利用不直接相邻图顶点的状态值依赖关系，从而进一步图顶点状态传递，最大化有效的数据并行度，使得图顶点状态传递能够充分利用底层并行计算资源并行有效地传递；并且，通层次化树型通信方式也缩短了通信时延，从而使得整体计算时间更少，加快图计算的整体收敛速度。因此，通过以上方式使得图计算收敛更快。
2. 减少数据访问量：本发明通过将加载到缓存的图数据块反复处理到局部收敛才加载下一个块进行处理，从而提高数据时间局部性，避免数据的反复加载；并且，通过依赖感知的异步执行方式避免无效图顶点处理，这也相应地减少了相应图数据的加载；同时，通过有效规则化图顶点状态更新，将图顶点状态的随机数据访问有效地转化为顺序的内存访问，保证很好的数据访问局部性，减少冗余的图数据访问，从而能够有效减少存储资源和内存带宽的消耗，并降低图顶点状态更新开销；此外，通过合并核心图顶点状态数据的存储和访问，提高数据空间/时间局部性，保证每次加载的一批数据（比如Tesla v100每次数据访问都会至少加载512字节的数据，传统CPU架构每次访问64字节的数据）大多数是有用的，并且能够在cache中被反复重用多次，从而极大地减少数据访问次数和访问量，提高核的利用率。因此，通过以上方式能够有效减少图结构数据和图顶点状态数据的访问量。
3. 降低网络通信开销：本发明所提供的层次化树型通信模式能够使得大量无规则的全局小消息通信规则化，即大多数无规则的全局小消息通信能够被转换成了局部通信，从而有效限制了大量无规则小消息通信对整个网络以及整个集群的负面影响；并且通过消息层次化通信方式来有效合并小消息，从而充分利用网络带宽等。而且，各计算节点的图数据以及其所管辖的计算节点的图数据被反复处理直到收敛后才和其他同级节点进行通信，从而可以极大地减少通信次数、消息数量和通信量，并且可以有效减少全局同步开销，降低了分布式图计算的通信开销，提高整个集群的计算资源利用率。同时，在消息发送前，对发往相同计算节点的消息进行合并，将非规则小消息通信转换成顺序的规则通信，从而充分利用网络带宽，减少频繁的小消息通信开销；此外，对通信消息进行压缩，通过去掉图顶点ID等不必要的信息来进一步降低通信量。

**附图说明**

图7为本发明高能效分布式异步图处理流程图。

图8为本发明层次化超大规模分布式通信流程图。

**具体实施方式**

为了使本发明的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅用以解释本方面，并不用于限定本发明。此外，下面所描述的本发明的各个实施方式中所涉及到的技术特征只要彼此之间未构成冲突就可以相互组合。

本发明提供的一种面向后E级超级计算机的高性能图计算技术，其核心思想包括：各计算节点利用细粒度的块来进行异步图处理，使得每个加载到缓存的块被处理到局部收敛才换出，提高数据访问局部性，避免数据冗余的反复加载；并且，优先地选择重要图数据块进行处理来加快重要图顶点的状态传播速度。其次，对每个块的处理，采用一种图拓扑感知的异步图处理机制，使活跃图顶点的最新状态能够沿着图拓扑快速传递，加快图顶点状态速度。对图计算的图顶点状态更新进行规则化来有效降低随机访存，提高cache和内存带宽利用率，降低图顶点状态更新开销；同时，合并核心图顶点状态数据的存储和访问，提高数据时间/空间局部性，保证每次加载到cache的图顶点状态数据大多都是有效的。生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系，以此来进一步加快图顶点状态的传递速度，提高计算资源的有效并行度。每个计算节点的每轮图计算需要将其图划分块计算到局部收敛为止才触发通信，加快了图顶点状态传递，充分挖掘各计算节点的计算能力，也减少了通信次数和数据量。在各个计算节点的每轮图计算后，通过基于分组的层次化的树型通信模式，使得大量无规则的全局小消息通信规则化，即大多数无规则的全局小消息通信能够被转换成了局部通信，从而有效限制了大量无规则小消息通信对整个网络以及整个集群的负面影响；并且通过消息层次化通信方式来有效合并小消息，从而充分利用网络带宽。在通信前，对发往相同计算节点的消息进行合并和压缩，将非规则小消息通信转换成顺序的规则通信，并去掉通信消息中的图顶点ID等不必要的信息，从而减少频繁的小消息通信开销，降低通信量，提高带宽利用率。

实施例所提供的一种面向后E级超级计算机的高性能图计算技术，其流程如图7和图8所示，包括高能效分布式异步图处理模块和层次化超大规模分布式通信模块。具体如下：

（1）高能效分布式异步图处理机制。其主要包括如下子步骤：

（1.1）各计算节点将其本地图分区划分为细粒度的图划分块*P*，图划分块的大小|*P*|由以下公式决定：|*P*| + \*|*V*|\*|| + |*R|* ≤ |*LLC*|，其中|*G*|为图数据的大小，|*V*|为图顶点的数量，||为每个图顶点状态值的大小，|*LLC*|为L3 cache的大小，|*R|*表示预留的空间。|*P*|的值为满足以上公式的最大正整数值。

（1.2）统计各划分块中图顶点的平均度数=，其中是图划分块P的图顶点集合，是图划分块*P*中包含的图顶点数量，是图顶点的度数；随后将根据阈值*Threshold*来区分高度数块和低度数块，*Threshold*是由用户定义的高度数块的比例来决定，例如：现有100个图划分块，用户定义的高度数块的比例为5%，则将图划分块按照降序排序后，*Threshold*为第5个图划分块的；*ADvg*大于*Threshold*的划分块被判定为高度数块，随后将选取的高度数块来构建核心子图。

（1.3）核心图顶点状态合并。首先根据用户自定义的核心图顶点比例来判定核心图顶点，例如，如果图中有10000个图顶点，用户自定义的核心图顶点比例为0.5%，则将图顶点按照其度数的降序进行排序后，前50个图顶点为核心图顶点；随后，如图2所示，每个计算节点将其本地的高度数图顶点状态合并存储在其本地的合并状态表中；为了快速访问到本地的核心图顶点状态值，各计算节点构建相应哈希表，其中每个*key-value*对为<*v*，>，为图顶点*v*的状态值在合并状态表中的地址。

（1.4）在每一轮图计算中，各计算节点选取包含活跃图顶点的图划分块进行处理，并且在选取时，优先选择核心子图中的图划分块，选取图划分块后进入步骤（1.5）。若该计算节点的所有图划分块均收敛（即，该计算节点的图分区中没有活跃图顶点），则表示当前轮图计算完成；如果此时达到了通信条件（即，该计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则触发分布式通信模块来同步不同计算节点上的图顶点状态；如果此时还没有达到通信条件，则该计算节点进入步骤（1.10）。

（1.5）在处理每一个选取的图划分块时，计算节点的每个CPU核选取该块中的一个活跃图顶点并压入其私有队列*Q*中（每个CPU核对应一个队列*Q*），以便以该图顶点作为源顶点来按照图拓扑获取图数据（以广度优先的顺序获取），并按照图拓扑顺序进行异步图计算；如果该图划分块中没有活跃图顶点，则表示当前选取的图划分块已经局部收敛，返回步骤（1.4）。

（1.6）每个计算节点的每个CPU核从其对应的队列*Q*中按顺序取出一个图顶点，然后获取该图顶点的邻居图顶点（与一起构成边）以及相应图顶点的状态值。

（1.7）对获取的图数据进行处理。将每条边的源顶点状态值传递到该边的目的顶点来更新目的顶点的状态值。同时，读取图顶点状态值依赖关系表，通过生成的图顶点状态值依赖关系，将通信后的图顶点状态值快速传递给其他图顶点。每次图顶点状态传递会生成一个图顶点状态更新，即<目的图顶点ID，更新状态值>。队列*Q*中所有图顶点的相关图数据被处理后，所生成的图顶点状态更新会生成图顶点状态更新流。

（1.8）对图顶点状态更新流按照其目的图顶点ID进行排序，然后将排序后的图顶点状态更新，顺序地应用到相应图顶点，即将更新状态值和目的图顶点的当前状态值进行计算，得出目的图顶点的最新状态值。对核心图顶点的状态更新，需要更新其在合并状态表的状态值，从而提高核心图顶点状态访问的空间/时间局部性。如果图顶点的状态值被更新，则该图顶点被设置为活跃图顶点，并压入队列*Q*中。

（1.9）判断队列*Q*中是否还存在活跃图顶点。若是，则返回步骤（1.6）；否则，返回步骤（1.5）。

（1.10）生成需要通信状态值的图顶点与其他不直接相邻图顶点之间的状态值依赖关系。具体来说，由于图顶点状态值之间的依赖关系通常是线性关系，因此，两个不直接相邻的图顶点状态值的依赖关系可以表示为：=*a*\*+*b*，和为两个不直接相邻的图顶点，和为图顶点和的状态值，*a*和*b*为常量。例如，在图2(a)中，需要通信图顶点状态值的图顶点为，对于SSSP算法，与其他不直接相邻图顶点（例如，）之间的状态值依赖关系如图2(a)所示，其中常量*a*和*b*分别为1和9。该依赖关系的计算方式为，进行两次迭代处理，分别得到两组图顶点和的状态值，即<, >和<, >，随后将根据这两组状态值计算出相应的常量*a*和*b*。

（1.11）图顶点状态值依赖关系存储。当两个不直接相邻的图顶点状态值的依赖关系被计算出来后，相应的参数（即，，，*a*和*b*）被存储在一个图顶点状态值依赖关系表中，如图2(b)。当每次通信后，需要通信状态值的图顶点被更新图顶点状态值后，可以利用生成的图顶点状态值依赖关系，将更新后的图顶点状态值快速传递给更多的图顶点，加快图顶点状态的传递速度和传播范围。

（1.12）在生成图顶点状态值依赖关系的过程中，如果某个时刻达到了通信条件（即，当前计算节点上的图数据和其管辖的计算节点上的图数据均收敛），则返回步骤（1.4）。

（2）层次化超大规模分布式通信。其主要包括如下子步骤：

（2.1）对计算节点进行分组，构建层次化的树型通信架构。如图1的示例中，每个分组包含4个计算节点（除树型结构根节点对应的计算节点外），随后各分组之间构成如图1所示的树型通信架构。各计算节点（例如，图1中的Node y）在通信时，只与自己所管辖的计算节点（树型结构的子节点，图1中的Node x，Node x+1，Node x+2和Node x+3）、自己所属的计算节点（树型结构的父亲节点，图1中的Node 5）和该所属节点所管辖的其余同层次节点（树型结构的兄弟节点，图1中的Node y+1，Node y+2和Node y+3）进行通信。

（2.2）对图数据进行分区，随后将图分区分配到各计算节点。具体来说，图顶点之间依赖紧密的子图（称作社区）将被分配到同一分组中，甚至同一计算节点，以此来降低全局通信开销。如图3所示，通过社区检测算法（例如，基于标签传播的社区检测算法）来检测出图结构中的社区关系，随后将具有社区相关的图数据分配到同一分组中（例如，将社区结构0分配给图1中的计算节点Node x，Node x+1，Node x+2和Node x+3）。此外，在进行分区时采用笛卡尔顶点切分方法，在该方法中，只有主代理图顶点同时拥有该图顶点的出边和入边，而代理图顶点只有该图顶点的出边或者入边的一类，如图4（a）所示。按照以上方式，图数据将被切分为图分区，然后各个图分区将分配到超级计算机的各个计算节点。

（2.3）每个计算节点（例如，图1中的Node *y*，位于第3层）在完成其第*Tj*轮图计算后，将需要通信状态值的图顶点的状态值更新生成消息。如图4(a)所示，Node *y*上的图分区收敛后，需要把其镜像代理图顶点（即图顶点，和）的状态值生成通信消息，以便发送到其他计算节点上相应的主代理图顶点（例如，Node *y+1*上的图顶点，和）来同步不同计算节点上相同图顶点的状态值。

（2.4）当通信消息生成以后，如图5(a)所示，对同一图顶点相关的通信消息进行规约，即将同一图顶点的多个通信消息合并为一个通信消息（例如，对于SSSP算法来说，需要计算出这些通信消息中得状态值更新最小值），从而减少通信量。

（2.5）如图5(b)所示，对发往同一计算节点的信息合并到同一队列以便统一发送，从而将不规则的小消息通信转化为规则的大消息通信，提高网络带宽利用率。

（2.6）对通信消息进行压缩。如图4(b)所示，将每个队列中的通信消息进行压缩。具体来说，通信消息根据图顶点ID进行排序，随后通过使用位表来标识图顶点状态值所对应的图顶点ID信息，将消息中的图顶点ID信息去掉，从而进一步减少通信量。

（2.7）该计算节点Node *y*所管辖的计算节点（即，图1中的Node *x*，Node *x+1*， Node *x+2*，和Node *x+3*，位于第4层）和其自身的图数据均收敛后，该计算节点Node *y*将规约、合并和压缩后的图顶点状态数据消息发送到其他计算节点进行通信。具体来说，多机间的通信行为主要分为以下三种行为：

①该计算节点Node y和其所管辖的计算节点（即，图1中的Node x，Node x+1， Node x+2，和Node x+3）之间进行相互通信。具体来说，在通信时，Node y所管辖的计算节点先将他们的通信消息发送给Node y，Node y对通信消息按照步骤（2.4）和步骤（2.5）所描述的方法进行规约、合并和压缩后，Node y再将规整后的消息分别发送给其所管辖的计算节点，从而避免各计算节点之间频繁的小消息通信。

②若按以上方式通信后，相应计算节点的图顶点状态值均不改变（即计算节点Node y和其所管辖的计算节点上的图数据整体达到局部收敛），则计算节点Node y将利用归总后的通信消息与其所属的计算节点（即，图1中的Node 5，位于第2层）和该所属节点所管辖的其余同层次节点（即，图1中的Node *y+1*，Node *y+2*，和Node *y+3*，位于第3层）进行通信，从而同步图顶点状态值。

③若该计算节点Node y和其所管辖的计算节点需要与不直接管理或不直接所属的计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）进行通信时，通信消息会根据树型通信结构进行发送和传递，避免全局的频繁小消息传递。如图6所示，相关通信消息会先发送到Node y进行汇集，随后按照树型层次化通信结构，将汇聚后的消息发送到Node 5，随后再发送到Node y+3，Node y+3最后将相关通信消息分别发送到相应计算节点（例如 ，图1中的Node n，Node n+1，Node n+2和Node n+3）。通过这样的方式，使得全局通信消息被充分规约、合并和压缩，使得网络通信行为更规则。

（2.8）当所有计算节点的图数据均收敛后，即当所有计算节点与其所属的计算节点和其所属节点所管辖的其余同层次节点进行通信后，图顶点状态值均不再改变，则计算结束。

本领域的技术人员容易理解，以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内所作任何修改，等同替换和改进等，均应包含在本发明的保护范围之内。

**说 明 书 附 图**



图7



图8