

CPU 和 GPU 混合集群的负载均衡策略

袁 壮, 陈庆奎, 曹剑炜

(上海理工大学 光电信息与计算机工程学院, 上海 200093)

摘 要: 为了充分利用 GPU 集群(Cluster)中各节点的资源,提高 GPU 集群(Cluster)整体的计算效率. 本文以 3G 网络中海量视频质量分析为研究背景,提出了一种面向 CPU 和 GPU 集群的负载均衡策略,构建了一个 GPU 异构集群系统. 实验表明,该负载均衡策略能很好地利用 GPU 和 CPU 计算资源、提高集群计算效率.

关键词: GPU 集群; 负载均衡; 计算资源; 计算效率

中图分类号: TP302

文献标识码: A

文章编号: 1000-7180(2012)09-0047-04

A Load Balancing Strategy of CPU-GPU Cluster

YUAN Zhuang, CHEN Qing-kui, CAO Jian-wei

(School of Optical-Electrical and Computer Engineering, University of Shanghai for
Science and Technology, Shanghai 200093, China)

Abstract: In order to making full use of the resource of each node of the GPU cluster, and improving the computing performance of the GPU cluster. In this article, Taking the mass video quality analysis in 3G network as research background, made a Load balancing strategy based on CPU-GPU cluster, and built a GPU heterogeneous cluster system. Through a large number of experiments show that the load balancing policy can full advantage of the GPU and CPU computing resources and improve the efficiency of cluster computing.

Key words: GPU cluster; load balance; compute resource; compute efficiency

1 引言

当今,海量数据的同步处理对计算能力要求越来越高.例如,3G 视频质量分析,其要先对 H264 的 NAL 流进行熵解码,它是一种 I/O^[1]密集型计算,对处理机的 CPU 和 I/O 能力有很高的要求;其次利用熵解码得出的预测系数和残差数据,对图像进行重构并得出图像质量,是数据密集型计算,要求处理机有很强的并行计算^[2]能力.同时,数据流的处理^[3]要求流上各节点之间的要有相匹配能力,以保证各处理机的计算资源有较高的利用率.

本文以 3G 网络中海量视频质量分析为背景,提出了一种基于 GPU 与 CPU 的负载均衡策略.构建了一个 GPU 集群,该集群有两种节点,一种用于

对 H264 视频的 NAL 流进行熵解码,一种用于对图像进行重构得出图像质量.同时对 CPU 与 GPU 计算能力上的差异做了深入研究,数量化了它们的处理能力,构建了能力匹配^[4-6]模型;利用 Agent 机制^[7-8]进行动态调整,优化能力匹配模型,使流上各节点的处理能力逐渐匹配,达到负载均衡^[9].

2 GPU 集群结构

H264 视频解码算法主要由两部分组成:(1)熵解码;(2)帧内预测.其中熵解码逻辑复杂度高,而帧内预测则属于密集型数值计算,因此传统的计算机集群很难实现 3G 网络中海量视频的质量分析.针对 GPU 与生俱来就具有众核并行处理器的特点,加之 CUDA 技术的发展,这里引入了 GPU 集群的

收稿日期: 2012-02-05; 修回日期: 2012-03-10

基金项目: 国家自然科学基金项目(60970012,61003031);上海信息技术领域重点科研攻关项目(09511501000);上海市重点学科建设项目(S30501)

概念,即将普通集群中的一部分节点用支持 CUDA 通用计算的 GUP 节点替代. 这样该部分节点就能很好地处理 H264 视频解码过程中数值计算密集部分.

该集群可划分为 4 大部分,如图 1 所示. 各部分的功能简介如下:

(1)逻辑计算密集型节点 LCN(Logic Calculation Note):该节点有较高的 CPU 配置,主要负责 H. 264 视频解码算法中的熵解码部分,无支持 CUDA 通用计算的 GPU.

(2)数值计算密集型节点 NCN(Numerical Calculation Note):该节点含有支持 CUDA 通用计算的 GPU、主要负责 H. 264 视频解码算法中帧内预测等密集型数值计算.

(3)高速交换机 SN(High-speed switches Note):负责中间数据的暂存与转发.

(4)主控机 MN(Master Note):该节点负责决策任务分配、资源调度、计算结果的监控与存储.

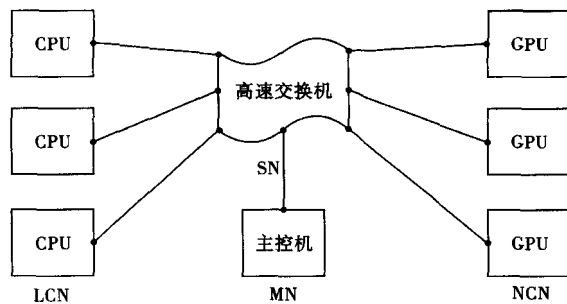


图 1 GUP 集群

这里将若干个(至少一个)LCN 作为一个功能单元,称之为 CPU 功能单元(CPU Function Unit, CFU). 同样将若干个 NCN 作为一个功能单元,并称其为 GPU 功能单元(GPU Function Unit, GFU). 系统运行时,视频流首先在 CFU 进行熵解码,以帧为单位形成中间数据通过 SN 转发作为 GFU 的数据输入,然后由 GFU 计算分析出视频质量.

3 负载均衡策略

3.1 能力匹配模型

一个视频图像通常由若干 16×16 的宏块组成. 这里以一个宏块为最小任务单位,对于一个分辨率大小为 $W \times H$ 的 H264 视频,设它的每一帧图像所含宏块数为 N ,则:

$$N = \frac{W \times H}{16 \times 16} \quad (1)$$

定义 $V_{Macroblock}$ 为处理机单位时间内处理的宏块数.

定义 V 为处理机单位时间内处理的图像个数,则:

$$V = \left[\frac{V_{Macroblock}}{N} \right] = \left[\frac{V_{Macroblock} \times 16 \times 16}{W \times H} \right] \quad (2)$$

以 CPU 功能单元和 GPU 功能单元为能力匹配对象,要求一组相对应的 CPU 功能单元与 GPU 功能单元有相当的处理能力. 即要求:

$$V_{CFU} = V_{GFU} \Rightarrow \sum_{i=1,2,3,\dots} u_i = \sum_{j=1,2,3,\dots} v_{g_j} \quad (3)$$

其中 u_i 表示 CFU 中第 i 个 LCN 的处理速度, v_{g_j} 表示 GFU 中第 j 个 NCN 的处理速度. 不过在系统的实际运行中等式(3)的左右两边往往很难相等,为此这里提出了一个最佳匹配算法.

首先构建逻辑计算密集型节点 LCN 的能力集合 VC 和数值计算密集型节点 NCN 的能力集合 VG,同时令集群中含 m 个 LCN 节点, n 个 NCN 节点.

$$VC = \{u_1, u_2, \dots, u_m, u_1 + u_2, u_1 + u_3, \dots, u_{m-1} + u_m, \dots, u_1 + u_2 + u_3, \dots, u_{m-2} + u_{m-1} + u_m\} \quad (4)$$

$$VG = \{v_{g_1}, v_{g_2}, \dots, v_{g_n}, v_{g_1} + v_{g_2}, v_{g_1} + v_{g_3}, \dots, v_{g_{n-1}} + v_{g_n}, \dots, v_{g_1} + v_{g_2} + v_{g_3}, \dots, v_{g_{n-2}} + v_{g_{n-1}} + v_{g_n}\} \quad (5)$$

由式(4)、(5)可以看出集群中的任一元素最多与 3 个节点有关,当然 $m, n \geq 3$,这样即保证产生足够多的组合种数,又大大降低了集群中元素个数,同时也大大降低了匹配的时间复杂度,节约了匹配时间.

然后找出集合 VC 与 VG 中大小最接近的两个元素 VC_i 和 VG_j . 用与 VC_i 相关的 LCN 节点构成一个 CPU 功能单元,用与 VG_j 相关的 NCN 节点构成一个 GPU 功能单元. 它们将构成相应的一对工作组. 然后再去除匹配过的节点,重新构建集合 VC 与 VG,找出下一个工作组. 如此重复,直到 VC 或 VG 中的元素为 0.

3.2 Agent 动态调整机制

定义 1 (处理机能力)处理机单位时间内处理的图像个数,用 A 表示,其值与处理速度 V 相等.

由 3.1 节可知,集群分为若干工作组 WG (Work Group),用 AWG_k 表示工作组 k 的处理能力,有

$$AKG_k = \min(ACFU_k, AGFU_k) \quad (6)$$

其中 $ACFU_k$ 表示工作组 k 中 CPU 功能单元

处理能力, $AGFU_k$ 表示工作组 k 中 GPU 功能单元处理能力. 显然 $ACFU_k$ 与 $AGFU_k$ 越接近, 整个集群的处理能力就越大. 但由于各节点的处理能力存在一定的变化, 使得当前分组并非最优, 因此需要动态地进行调整, 得出最优的组合. 为此在每个节点中植入一个 Agent, CPU 功能单元节点的 Agent 命名为 AC, GPU 功能单元节点的 Agent 命名为 AG, 主控机节点的 Agent 命名为 AM. AC、AG 实时对相应节点进行监控. 这里把节点的状态分为三种:

$$Status = \{1, 0, -1\} \quad (7)$$

Status=0 时表示该节点负载均衡, 不做任何处理; Status=1 时表示该节点超载; Status=-1 时表示节点有能力处理更多的任务.

当 Status=1 时, 说明该节点超载, 该处理机将回馈消息 Overload, 其中 Overload 有两个等级:

$$Overload = \{1, 2\} \quad (8)$$

当 Overload=1 时, 表示超载程度轻微, 对该节点下调 1 个单位能力值; 当 Overload=2 时, 表示超载程度严重, 对该节点下调 2 个单位能力值.

当 Status=-1 时, 表明处理的任务量低于自身的处理能力, 该节点处于“饥饿”状态. 自身将根据饥

饿值 Hunger 进行调整, 同样 Hunger 有两个等级:

$$Hunger = \{1, 2\} \quad (9)$$

当 Hunger=1 时, 表示处理能力轻度过剩, 自身上调 1 个单位的能力值; 当 Hunger=2 时, 表示处理能力严重过剩, 自身上调 2 个单位的能力值.

AM 将对整个过程进行监控, 当各节点의 Status 都为 0 时, 表明集群处于稳定状态. 反之, 将以最新的能力值按 3.1 节中的能力匹配模型重新进行分组.

4 实验结果与分析

4.1 实验环境

本实验集群的节点由 10 种不同配置的处理机构成, 其中 LCN 节点和 NCN 节点各 5 种. 实验数据为若干 H264 视频的 NAL 流, 通过抓包机从网络中抓取并存放各 LCN 节点的本地磁上. 3G 网络中的 H264 视频一般分为三种规格: 176×144 、 320×240 和 352×288 . 本次实验数据选用的是规格为 176×144 的视频(其他两种规格的视频亦可). 表 1 列出了各种处理机相应配置以及相应的处理能力.

表 1 各种处理机配置表

	CPU	GPU	Memory	NIC	OS	Ability(个/s)
LCN	Intel Core2 Duo	—	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	1200
LCN	Intel i3-530	—	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	1600
LCN	Intel i5-750	—	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	2300
LCN	Intel i7-870	—	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	2700
LCN	Intel i7-960	—	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	3200
NCN	Intel Core2 Duo	NVIDIA GTX260	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	1800
NCN	Intel Core2 Duo	NVIDIA GTX280	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	2000
NCN	Intel Core2 Duo	NVIDIA GTX295	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	2200
NCN	Intel Core2 Duo	NVIDIA GTX460	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	3000
NCN	Intel Core2 Duo	NVIDIA GTX480	Kingston 4GB DDR3	1000Mb/s	Ubuntu10.04	3300

4.2 实验结果

4.2.1 能力匹配模型的收敛性

本实验目的在于验证不同节点数量下该模型的收敛性. 实验发现, 不同规模的集群经过 3 次匹配后系统会达到一个稳定的状态. 图 2 和图 3 分别显示了 10 和 20 个节点下的收敛效果. 其中横坐标表示调整次数, 纵坐标表示集群的总的处理能力.

4.2.2 扩展性

本实验的目的在于研究集群的可扩展性. 由实

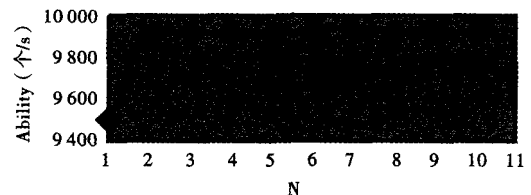


图 2 10 Nodes 时匹配模型的收敛效果

验结果可知, 集群的总的处理能力随着节点数目的增加呈线性增长趋势, 说明该方法具有高可扩展性,

易于集群的扩张. 如图 4 所示.

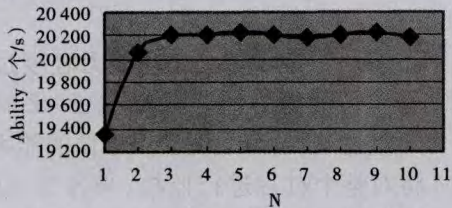


图 3 20 Nodes 时匹配模型的收敛效果

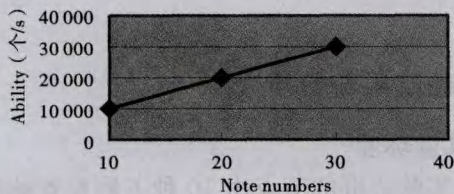


图 4 可扩展性

4.2.3 资源利用率

在做上面实验的同时,对 LCN 节点的 CPU 利用率和 NCN 节点的 GPU 利用率进行了统计,如图 5 和图 6 分别显示了 LCN 节点和 NCN 节点在不同节点数目下的平均利用率. 由图可知它们都在 90% 以上,表明资源利用率很高.

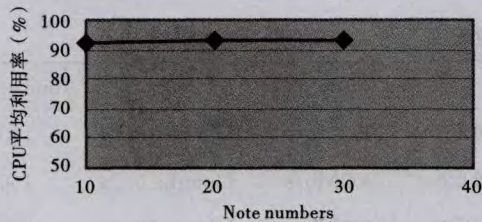


图 5 LCN 节点 CPU 平均利用率

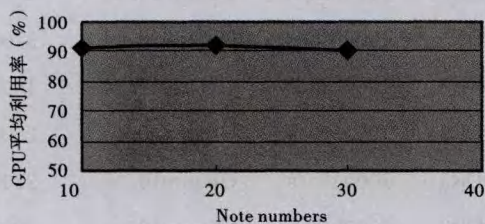


图 6 NCN 节点 GPU 平均利用率

5 结束语

通过构建能力匹配模型,灵活地利用了不同资

源构建 GPU 集群. 并通过 Agent 动态调整机制,进一步对集群进行调整,使集群中 CPU 功能单元与 GPU 功能单元拥有相匹配的处理能力,保证了视频流处理流上上下下两节点有相同的负载,即实现了它们之间的负载均衡. 最大程度地减少了资源的浪费,提高了资源利用率.

参考文献:

- [1] Erich Schikuta, Heinz Stockinger. Institute for applied computer science and information systems department of data engineering [R]. University of Vienna Ralhausstr. 19/4, A-1010 Vienna, Australia.
- [2] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报, 2004, 15(10): 1493-1504.
- [3] 曹锋, 周傲英. 基于图形处理器的数据流快速聚类[J]. 软件学报, 2007, 2(18): 291-302.
- [4] 罗洋, 曾国荪. 基于本体语义的网格服务能力匹配算法[J]. 计算机应用, 2004, 9(24): 52-54.
- [5] Zoltán Horváth, Manfred Liebmann. Performance of CFD Codes on CPU/GPU Clusters[C]// AIP Conf. Proc. 2010, 128(1): 1789-1792.
- [6] 方阳, 郑焱, 李俊, 等. 基于内容分组与能力匹配的邻居选择算法[J]. 计算机仿真, 2009, 26(4): 158-161.
- [7] Punam Bedi, Harmeet Kaur. Clustering recommendations to compute agent reputation [J]. Theory and Applications, 2005(1): 131-137.
- [8] Fukang yin, Fengshun Lu, Xiaoqun Cao, et al. GPS forward model computing study on CPU/GPU co-processing parallel system using CUDA [C]// Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing. China, Shanghai, 2010.
- [9] 叶荣华, 金芝, 王璞巍, 等. 一种需求驱动的自主 Web 服务聚集方法[J]. 软件学报, 2010, 21(6): 1181-1195.

作者简介:

袁 壮 男, (1987—), 硕士研究生. 研究方向为并行理论.
陈庆奎 男, (1966—), 教授、博士生导师. 研究方向为并行理论与并行数据库、网络计算.