清华大学本科生综合论文训练答辩

多视点(Multi-view Coding)视频的并行实时解码

本科生: 卿培

学号: 2006011291

指导老师: 孙立峰 副教授

@click

- 任务概述 & 任务目标
- MVC解码器性能优化
- 3D播放器设计与实现
- 存在的问题 & 未来工作方向

outline

@click

- ➡ 任务概述 & 任务目标
- MVC解码器性能优化
- 3D播放器设计与实现
- 存在的问题 & 未来工作方向

to begin with intro and aim

@click

任务概述

- ✦基于现有的MVC Decoder进行性能优化
- ✦基于NVIDIA 3D Vision眼镜实现有3D效果的播放器

system composition – the last 2 modules

decoder optimization

&

3d player design

@click

任务目标

- ✓ 修正MVC解码器算法，使得解码结果与参考软件一致
- ✓ 优化解码器性能，CPU实时解码720x576两路视频
- ✓ 设计实现一个3D播放器

debug

real-time decoding

3d player

all the listed has been done and we are continuing to show the realization

@click

✓ 任务概述 & 任务目标

➡ MVC解码器性能优化

- 3D播放器设计与实现
- 存在的问题 & 未来工作方向

continue to show the optimization

@click

MVC Decoder性能优化

- 性能分析
- 单核优化
- 并行优化
- 实验结果

schedule:

performance analysis
->
single core optimization
->
parallelization
->
test

@click

性能分析

- Visual Studio 2008 - Analysis -> Profile
- Intel VTune Performance Analyzer

在编译时加入一些辅助代码,运行时通过这部分代码来标志进入 和推出一个函数的时间,借助这些隐藏的输出信息得到总运行时间在各个 函数内部的分布。

@click

Visual Studio分析结果

表 3.1　VS2008 性能分析报告（前 10 项记录）

Function Name	Exclusive Time①	Exclusive Time %	Number of Calls②
macroblockPredGetDataUV	341.01	14.41	2,632,478
macroblockPredGetDataY	321.37	13.58	1,316,239
idct4x4_c	316.86	13.39	5,028,960
macroblockGetPred_axb	181.29	7.66	1,316,239
macroblockGetPred_axb_Bi	133.42	5.64	571,944
macroblockGetHalfPel	132.47	5.6	148,552
Filter	101.92	4.31	6,215,308
addMacroblockdata	85.36	3.61	184,171
iquant4x4_c	84.48	3.57	5,028,960
macroblockPBPrediction	70.95	3	184,171

① 不包括函数体内调用其他函数的净运行时间
② 函数被调用次数

top 10 of vs profiling results

@click

VTune分析结果

表 3.2　VTune 性能分析报告 (前 10 项记录)

Function	Calls	Self Time	% in function
macroblockPredGetDataUV	2632478	774863	0.61
macroblockPredGetDataY	1316239	702575	0.58
idct4x4_c	5028960	398076	1
macroblockInterDecode16x16_y	184171	328834	0.46
macroblockGetPred_axb	1316239	311730	0.11
FilterMB	210600	203506	0.21
Filter	6215308	191937	1
iquant4x4_c	5028960	157459	1
macroblockInterDecode_uv	184171	154697	0.48
macroblockPBPrediction	184171	150609	0.05

top 10 of vtune profiling results

@click

优化方案

- 重写函数逻辑
- 循环的优化
- 汇编优化
- 并行优化

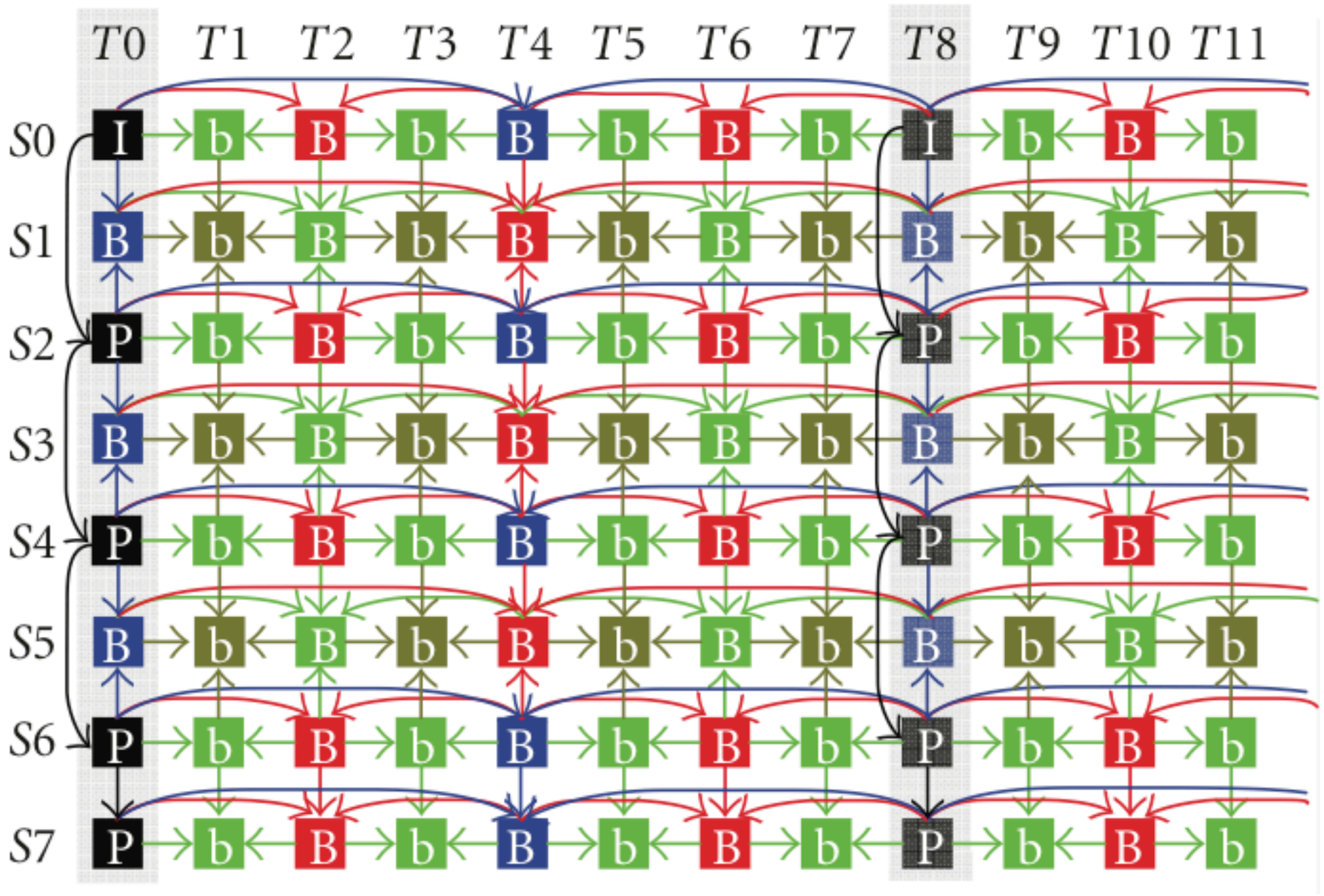
重写函数逻辑 解码器中有一些函数的逻辑判断有多处重复,降低了性能。如果能够理清楚函数逻辑,修改函数内部执行的顺序,就能够加快一些速度。

循环的优化 对帧和宏块的处理常常有两层for循环便利一个width×height的像素矩阵,逐像素进行操作。对于这样的函数,首先考虑是否能够将多次循 环内的操作合并到少量循环,再考虑循环结构对 cache 的友好程度。

汇编优化 在项目讨论中,罗翰先生提出 ffmpeg 等开源的视频项目中可能会有 一些变换的汇编优化版本,如果能够应用汇编优化,那么这些函数的性能 将会有大幅度的提升。就此,我们考虑对一些函数进行汇编优化。

@click

并行优化



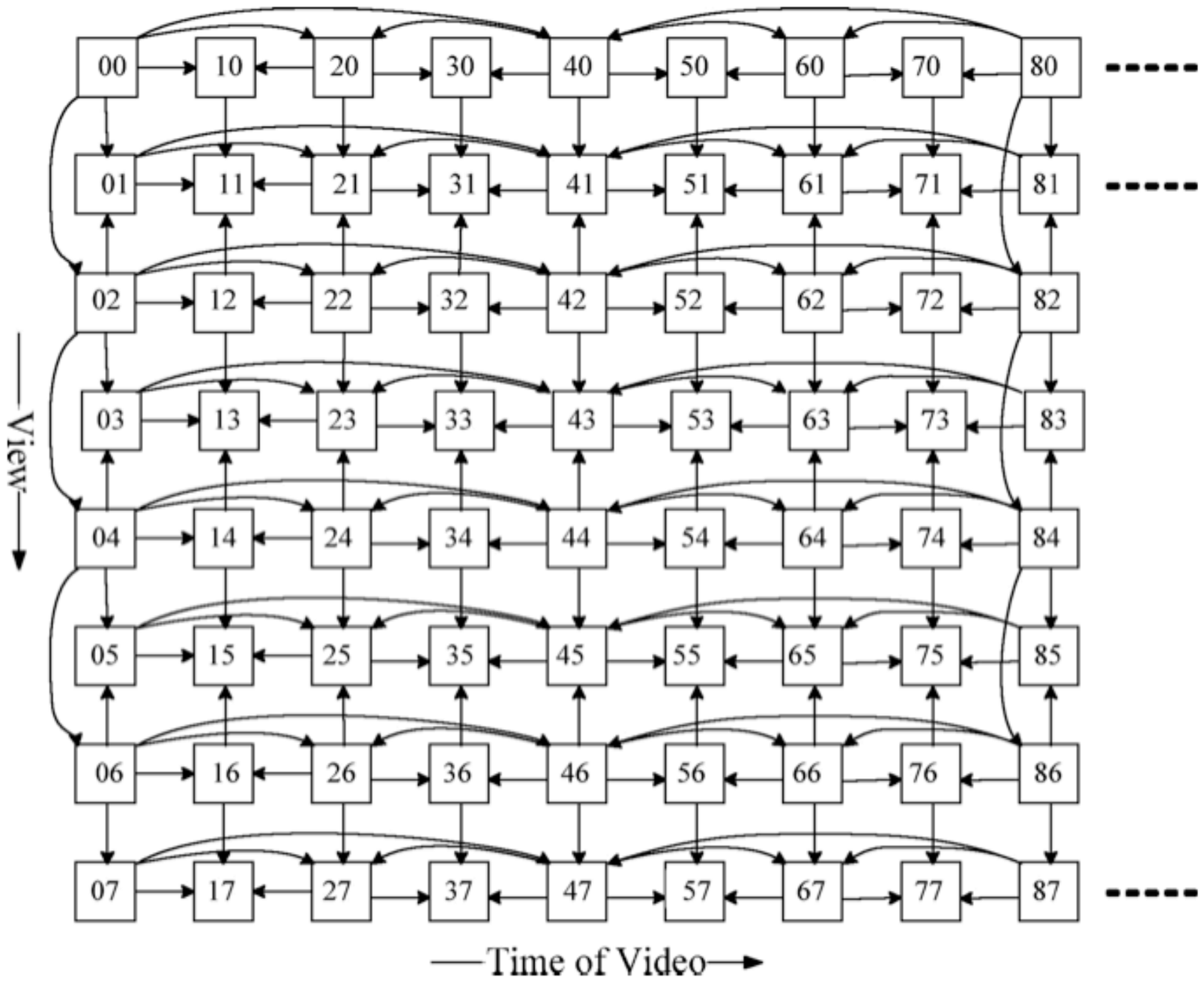
解码其中的 I 帧时,不需要参考任何帧,只要 I 帧的NAL Unit全部传入就可以开始对该帧解码。而对其中的 P 帧和 B 帧,都有箭头指向该帧,表示对这一帧的解码需要参考别的帧。

我们用 (Sn,Tn) 表示在 Tn 时刻, Sn 这一路码流对应的帧。
(S0,T1) 就需要参考 (S0,T0) 和 (S0,T2)
(S0,T2) 又要参考 (S0,T0) 和 (S0,T4)
(S0,T4) 则要参考 (S0,T0) 和 (S0,T8)

每一帧的解码都必须走过一条路径才能满足其参考帧都已经完成解码的条件。
这样就形成了一个分层的图,从一个空的根节点向下,第 i 层的节点是第 1 到第 i-1 层解码完毕后立刻可以解码的帧。

@click

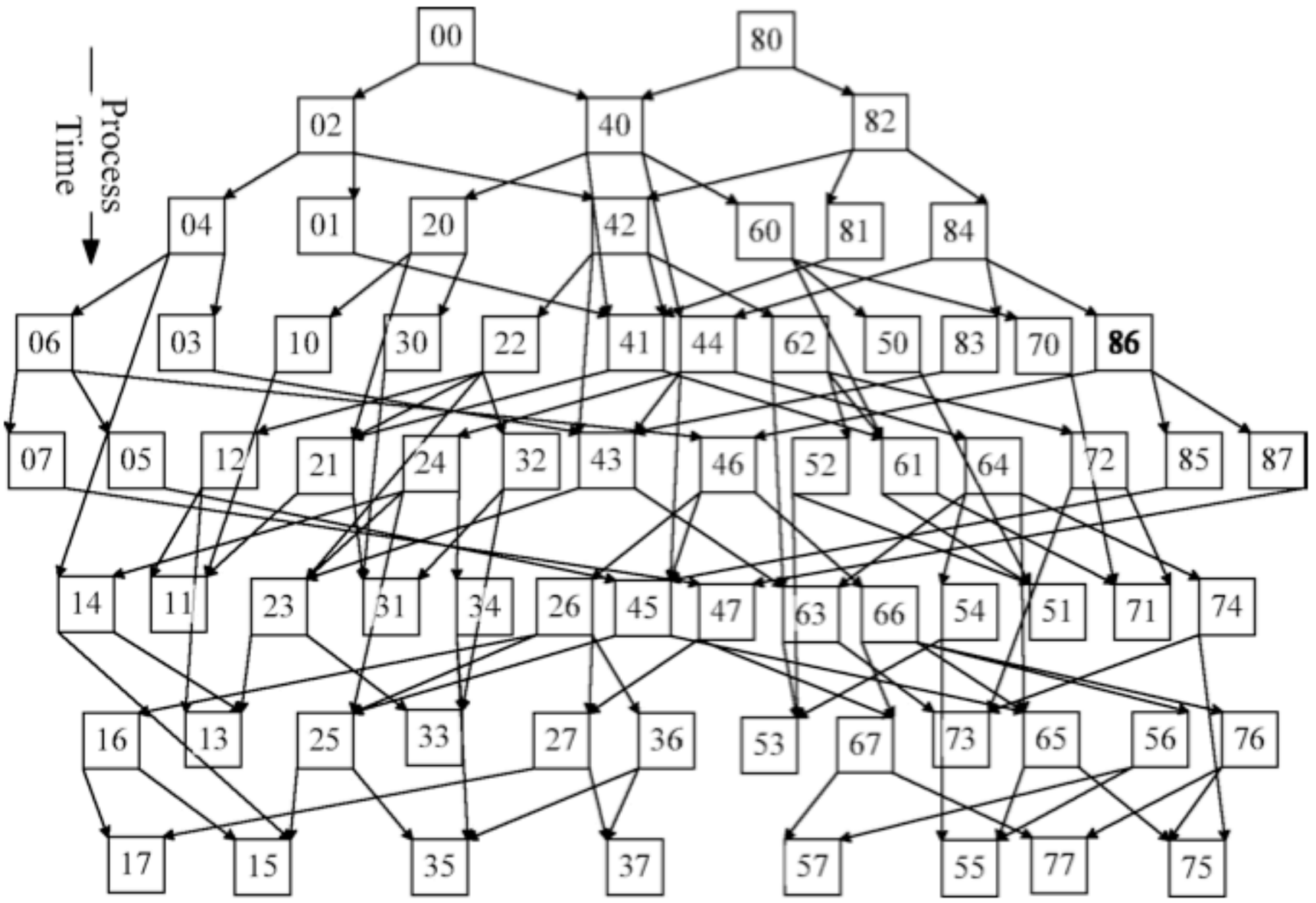
并行优化



for this particular prediction structure

@click

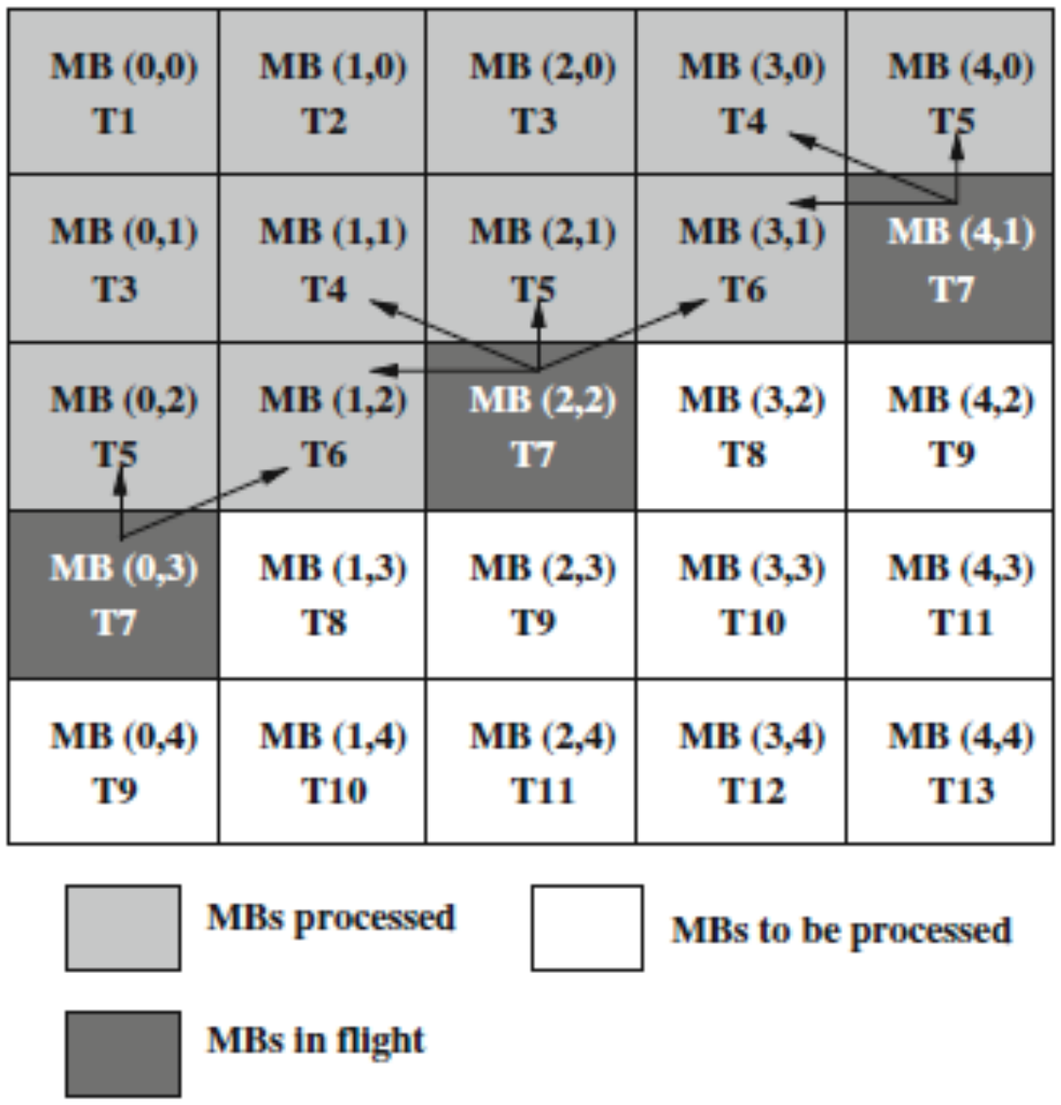
并行优化



this is the computed queue of parallel tasks

@click

并行优化



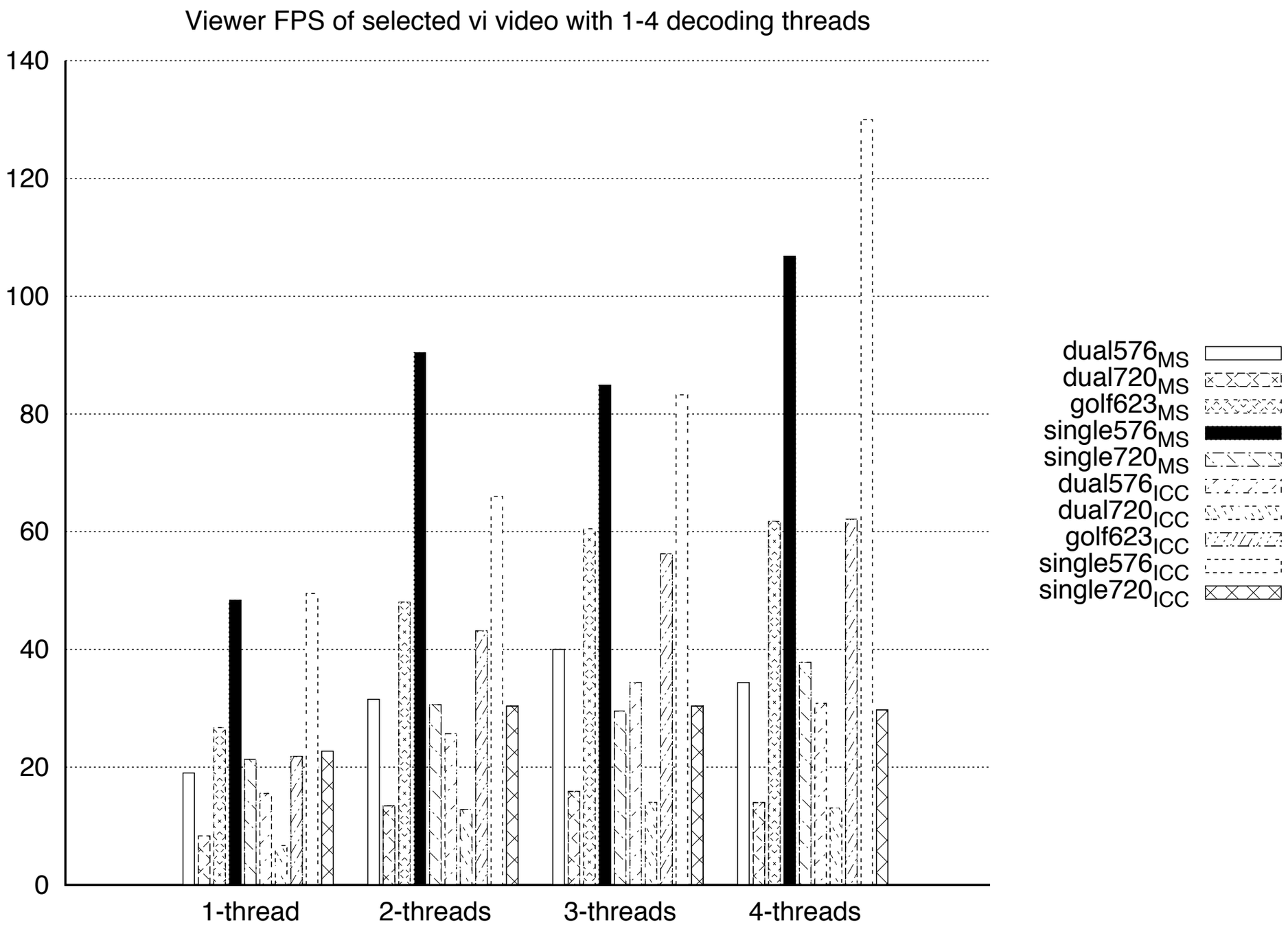
在解码 MB(4,1) 的时候,需要参考 MB(3,0)、 MB(3,1)、 MB(4,0) 的解码结果。

从 T1 时刻解码左上角 MB(0,0) 起
T2 时刻可以解码 MB(1,0)
T3 时刻可以同时解码 MB(2,0) 和 MB(0,1)
T4 可以同时解码 MB(3,0) 和 MB(1,1)
以此类推。

如果硬件资源足够同时解码3个宏块,那么全部16个宏块只要13个时间片就可以解码完成。

@click

实验结果（解码帧率）

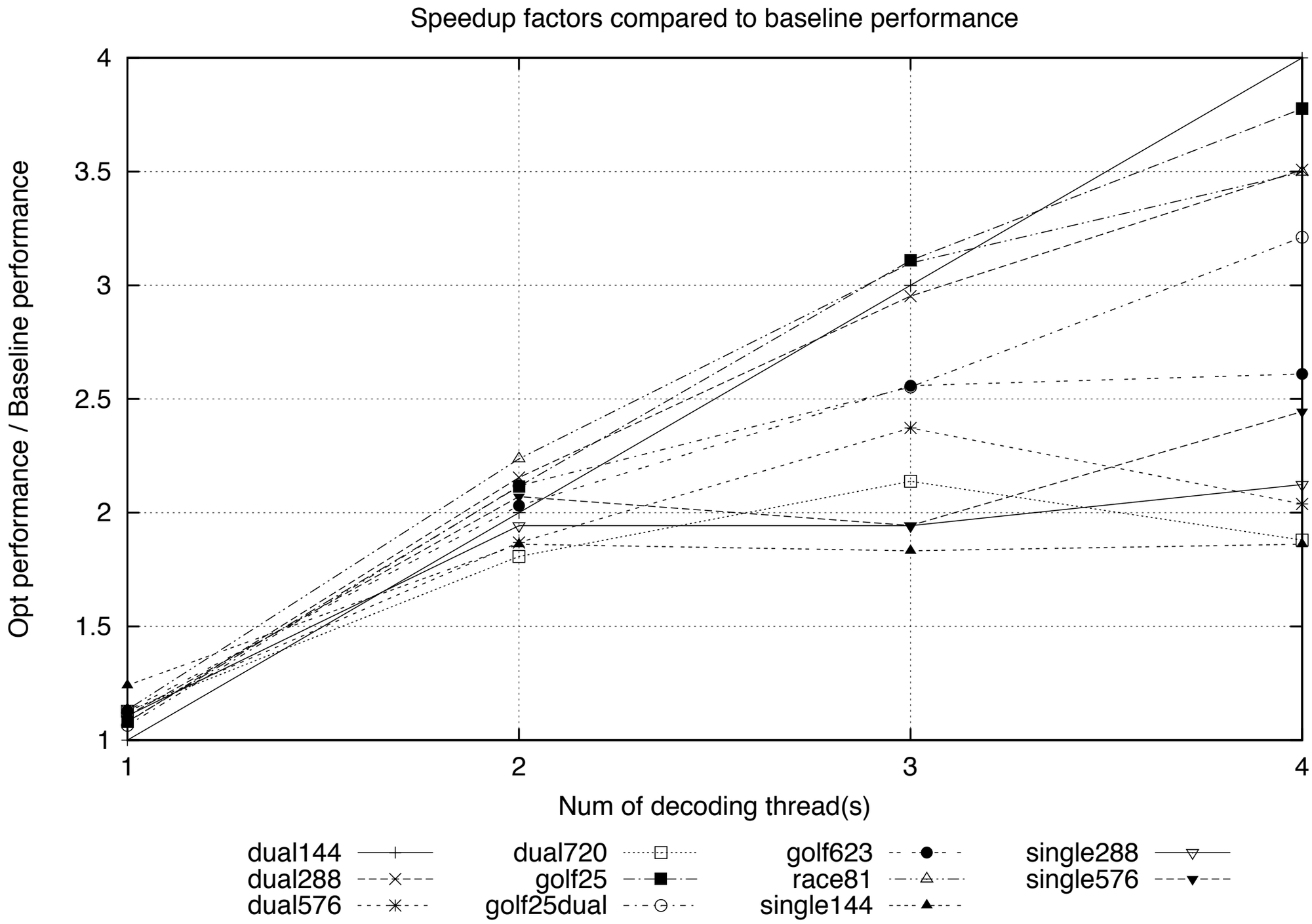


这里计算的帧率是同时解码 N 路时的帧率,也就是观看者实际看到的等效帧率。对两路视频而言,要达到 30fps 的等效帧率,必须能够实际解码速率达到 60fps。

从图中可以看到,对于实验目标针对的 720 × 576 的两路视频,只要线程数超过一个,就已经达到并超过了设定的实验目标。

@click

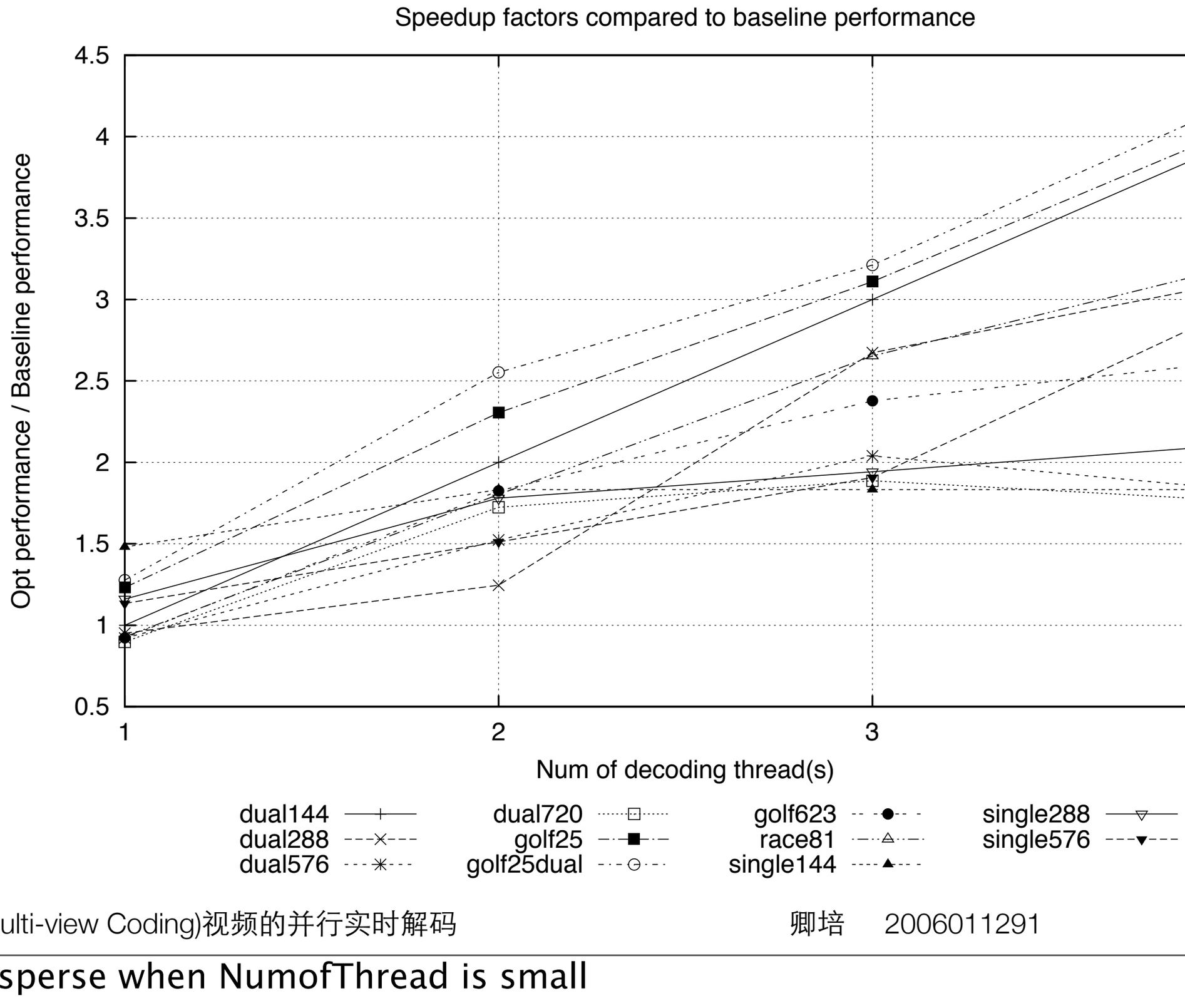
实验结果（MS并行加速比）



- dual144 的加速比正好为线程数,这是一个巧合。不过也能部分说明当视 频的并行性足够时,增加的线程资源能够被充分利用。
- golf25、golf25dual、dual288 和 race81 这四个视频也能充分利用额外的资 源。
- single144、single288、single576、single720 这些单路的视频加速比都没有 超过 2.6,这是由于一路视频的参考结构比较简单,没有视角间的参考。 调度器很难安排出能够并行处理的大于 2 个任务。
- golf623 的加速比低于预期,我们推测原因在于编码时的 GOPSize 设成 4 对于 8 路视频偏小了,造成并行程度不够。

@click

实验结果（ICC并行加速比）



more disperse when NumofThread is small

better parallel performance

@click

- ✓ 任务概述 & 任务目标
- ✓ MVC解码器性能优化
- ➡ 3D播放器设计与实现
- 存在的问题 & 未来工作方向

continue with the 3d player

@click

3D播放器设计与实现

- 硬件平台
- 可选方案
- 具体实现

hardware platform

possible solutions

realization details

@click

硬件平台

- NVIDIA3D Vision
  - 快门式眼镜
  - 120Hz显示器

NVIDIA 3D Vision眼睛与我们在电影院观看3D电影时使用的偏振式的眼镜不同，它是快门式的。左右眼两个镜片分别内置了一个类似于相机快门的装置，可以遮挡住视线。

120Hz刷新率的显示器每秒交替显示60张左眼可见的图像和60张右眼可见的图片，眼镜通过红外线同步装置发射的同步信号来控制两个快门，保证双眼看到对其渲染的不同画面。

3D效果的合成依靠的是我们万能的大脑~

@click

可选方案

- 查询NVAPI

  ✘没有相应API Function

- 以2xFPS播放+Enable3D程序运行

  ✘但见重影不见立体效果

- 借助Direct3D

  ✓成功看到3D效果

设计的总体思路就是先实现一张静态3D图的显示，然后由此扩展到每秒n张图的显示，就实现了视频的播放。

@click

因为要调用NVIDIA的硬件，我第一个想到的就是去查NVAPI，其中的确有Stereoscopic3D子项@click但是我只看到了一些状态查询、保存当前图像之类的函数，却没有生成画面用于显示的函数。NVIDIA对于注册开发人员还有一份更完整的API文档，可惜我的申请没有得到回应。

@click第一个途径失败后，我想借助NVAPI中提到的一个打开3D功能的函数来实现。希望我在以2xFPS显示图片序列前调用该函数，NV的驱动能够自动识别我的目的，用3D的方式显示。@click不过这个方法依然没有成功。

@click经过两周不断地google、浏览开发论坛的帖子之后，我在mtbs3d.com的论坛上找到一个成功的例子，该贴描述了一个借助D3D来渲染3D画面的方式，给了几个关键的设置说明，跟帖中也有表示用该方法成功的@click于是我就尝试了这个方法，终于成功。

具体实现

D3DXLoadSurfaceFromFile

D3DXLoadSurfaceFromFile


```
#define NVSTEREO_IMAGE_SIGNATURE 0x4433564e //NV3D
```


该方法简单说明如下， @click建立一个用于渲染的表面，D3D中为一个IDirect3DSurface9。

这个表面的宽度为2*imgWidth。 这个表面的高度为imgHeight+1。此为第一个magic number……

之后以左上角为原点load左眼图像，@click

将原点右移imgWidth个像素，再load右眼图像@click。

在渲染前还要写一个LPNVSTEREOIMAGEHEADER，设置宽度为2*imgWidth，高度为imgHeight，色彩深度为32bit，@click最神奇的是还要设一个signature。这个magic number我在其他地方没有看到过……

在调用Direct3D的renderer渲染之后，我终于看到了3D效果的静态图。

@click

具体实现

显示效果就是这样，直接看显示器会看到重影。透过眼睛看到的就是3D的场景。

@click

具体实现

- Frame rate cap
  - Accurate timer
    - QueryPerformanceCounter & QueryPerformanceFrequency
      - Retrieves the frequency of the high-resolution performance counter, if one exists. The frequency cannot change while the system is running. Link to this doc
  - Fn=(tn-t0)*FPS
  - Frame skip

@click在显示了静态图之后，我开始实现以一定帧率来显示图片。

@click这需要一个精确的计时器，getTicker函数是ms级的，比较粗糙。@click我使用的是Windows API提供的QueryPerformanceCounter与QueryPerformanceFrequency两个函数配合，可以做到us级的计时，可以满足一般视频的播放要求了。

On some CPUs, QueryPerformanceCounter() reads the CPU clocks, which causes it to be broken on speedstep CPUs. There's a patch you can download for AMD processors that fixes that. The value returned from QueryPerformanceFrequency should not change while the system is running. Link to this page

@click在播放第0帧之前，记下当前时间t0，在此后的某个时刻，获取当前时间tn，(tn-t0)*FPS就是该时刻应该显示的帧的序号。

@click视频播放还有个跳帧的要求，当显示速度跟不上视频的帧率时，应该跳过一定数量的帧来保证视频的帧与时间轴仍旧是对应上的。我在上述计算下一个渲染对象的时候，就隐含了跳过一些帧的机制。

具体实现

Algorithm 1 Frame rate cap

```
1:  _start ⇐ initialTime
2:  buffered = FALSE
3:  lastRenderedFrame ⇐ −1
4:  while TRUE do
5:     _current ⇐ currentTime − _start
6:     frameToRender ⇐ _current ∗ FPS
7:     if frameToRender + 1 > totalFrames then
8:        break {break when reaching the last frame}
9:     end if
10:    if buffered = FALSE then
11:       load Frame_{frameToRender+1} to memory
12:       buffered ⇐ TRUE
13:    end if
14:    if frameToRender ≠ lastRenderedFrame then
15:       render Frame_{frameToRender}
16:       lastRendered ⇐ frameToRender
17:       buffered ⇐ FALSE
18:    end if
19: end while
```


我们实现的帧率上限控制的机制如下:

在开始播放第 0 帧的时刻记下 当前时间 t0。
在渲染完第 i 帧 Fi 之后,获得当前时间 tn,
然后准备进行第 (tn −t0)×FPS 的渲染。

通过比较当前已渲染的帧 Fi 和待渲染的帧 Fn 是否一致 来决定是否进行渲染操作。

这个机制的核心就是

Fn =(tn−t0)×FPS

其中 FPS 为根据播放内容而预设的一个帧率,
tn 为当前时刻,
t0 为播放视频第0 帧的起始时刻,
Fn 为当前时刻应该渲染的帧的序号。

@click

实验结果

- 功能

  ✓有3D效果

- 性能?

  ✘21fps

- 瓶颈

  ➡D3DXLoadSurfaceFromFile()

@click功能@click有3D效果@click性能@click21fps@click瓶颈@clickAPI

- 我们首先考虑是否因为图像序列的格式为JPEG,在读取的时候需要时间解码。于是将图像序列转化为BMP格式的位图,测试发现帧率进一步降 至14fps左右。测试用图存为JPEG时每帧平均大小50KB,BMP则超过300KB。
- 有了JPEG和BMP的对比,我们认为磁盘I/O可能是一个瓶颈。对此,我们划分了500MB 的内存作为一个 RAMDISK。用HD Tune 测试这个分区的读写速度和随机访问时间等指标如下:

  Transfer Rate (Min) 2751.7MB/sec
  Transfer Rate (Max) 3363.2MB/sec
  Transfer Rate (Avg) 3239.8MB/sec
  Access Time (Avg) 0.0ms
  Burst Rate 3670.7MB/sec
  CPU Usage 0.0%

  将图像序列存放在该 RAMDISK 分区上再次进行测试,帧率依然在 21fps 左右。所以磁盘 IO 被排除在瓶颈之外。
- 此时我们决定用性能分析工具分析我们的程序,找出其中时间耗费最多的部分,将该部分打散成多个小的函数再次分析。如此循环了几次,终于找到性能瓶颈所在:我们调用 D3D 中的 D3DXLoadSurfaceFromFile() 函数两次,分别将左右眼的图像绘制到渲染的表面上。而这两个函数调用占用了整个程序运行总时间的 95% 以上。

✓ 任务概述 & 任务目标

✓ MVC解码器性能优化

✓ 3D播放器设计与实现

➡ 存在的问题 & 未来工作方向

continue with remaining problems and future work

@click

存在的问题

- 解码器
  - 多线程解码视频时的加速比不够稳定
  - 对 MVC 标准的支持尚不完整
  - 八路视频解码尚不能实时
- 3D播放器
  - 图像序列以磁盘文件形式存储
  - 帧率达不到流畅要求

@click
目前的解码器主要存在以下问题:
1.多线程解码视频时的加速比不够稳定。对于部分视频会出现增加线程反而降低解码性能的情形。
2.对 MVC 标准的支持尚不完整。
3.对一些裸眼立体显示设备要求的八路视频解码性能还达不到实时。

@click
3D 播放器主要存在以下问题:
1.播放的图像序列以磁盘文件而非内存中一段数据的形式存在,将来可能成为性能瓶颈。
2.播放时的帧率达不到流畅观看的要求,这主要是使用的一个 D3D API 函数调用造成的瓶颈。

未来工作方向

- 解码器
  - 多线程解码视频时的加速比不够稳定☞智能寻找最优线程数
  - 对 MVC 标准的支持尚不完整☞增加Main Profile和Stereo High Profile支持
  - 八路视频解码尚不能实时☞借助GPU进行transform运算
- 3D播放器
  - 图像序列以磁盘文件形式存储☞内存数据、对外接口
  - 帧率达不到流畅要求☞实现自行填充surface

@click

针对前文提出的问题,解码器在将来还需要进行以下工作:

1.对于加速比反常下降的视频,我们希望实现一个智能的判断机制,在不超 过线程上限的范围内,自动使用性能最优的线程数进行解码,避免性能损失。
2.增加对 MVC 标准中 Main Profile 和 Stereo High Profile 的支持,增强解码器的通用性。
3.借助 GPU 进行部分计算,实现两路高清视频的实时解码和八路视频的实时解码。

@click

3D 播放器尚需要实现以下改进:

1.自行实现对渲染表面的填充函数,替换 D3D API 调用,突破现有的性能 瓶颈,保证播放的画面流畅。
2.对播放器的输入进行修改,使其满足一个自定义的接口,方便将播放器和 解码器的输出对接,做到实时解码并播放。

@click

主要参考文献

I. Pang Y, Sun L, Wen J, et al. A framework for heuristic scheduling for parallel processing on multicore architecture: a case study with multiview video coding. IEEE Transactions on Circuits and Systems for Video Technology, 2009, 19(11):1658–1666

II. Smolic A, Müller K, Stefanoski N, et al. Coding algorithms for 3DTV—a survey. IEEE transactions on circuits and systems for video technology, 2007, 17(11):1606–1620

III. Merkel P, Smolic A, Müeller K, et al. Efficient prediction structures for multiview video coding. IEEE Transactions on circuits and systems for video technology, 2007, 17(11):1461–1473

Q&A

谢谢!

卿培
FIT 1-512
edwardtoday@gmail.com