Collection Performance Computation

25-April-2001
Outline

- Introduction
- References
- Stellar Distribution
- Star Processing Flow
- Processing Requirement
- Processor Capability
- Performance Prototype
- Summary
Introduction

- **Mission figure of merit**
  - Number of stars imaged during mission
- **A driving requirement for the FAME instrument** is the number of stars per second that can be collected.
- **Potential bottlenecks**
  - Downlink capacity
    - Bandwidth: 300 kb/s
    - Number of bits per star
  - Flight CPU performance
    - Window determination
    - CCD command generation
- **This presentation considers on-board computation**
• **Design drivers**
  ➤ Maximum stellar density per update period
  ▪ Efficiency of accessing star catalog
  ▪ Computation requirements per star
  ▪ CPU performance
  ▪ Star collection packetizing overhead
References

• Gaume
• Kaplan
• Monet
Stellar Distribution
Stellar Density

• Average number of stars per second is ??

• Maximum number of stars per second is ??

• Over sufficiently long periods the number stars in the FOV will tend to average number of
  - Non-galactic roll periods to rapidly tend to the average
Stellar Distribution
Stars in FOV

% of Double Apertures

Number of Stars

Date Review
Cumulative Numbers

Cumulative Double Apertures

Number of Stars
Analysis of Star Processing
Star States

- **Non-candidate Star**
  - In catalog but not near current FOV in current period
- **Candidate Star**
  - Close to field of view during current period
- **Field of View (FOV)**
  - In FOV during current period
- **CCD**
  - On a CCD during current period
Star States

Candidate Stars

Non-Candidate

FOV

CCD
Science Mode Dataflow

- Determine New Stars in FOV
- Determine Star Windows
- Generate CCD Commands
- CCD Control
- CCD Data

Legend:
- 1 Hz
- <1 Hz

- Refine Attitude & Rate
- Determine TDI Rate
- Centroid Windows

- TDI Period
- Guide Star Window Blocks

- Form Window Blocks
- Ground Processing

- Attitude & Rate
- Guide Stars
Determine Candidate Stars

- Tile 101
- Tile 217
- Tile 102
- Tile 218

FOV
Direction of Rotation
Bounding Box

FOV Motion during Current Period

Date
Review
15
Determine Candidate Stars

Method 1:
For each Aperture
  For each subdivision of the Update Period
  
  \{
    Generate aperture vector
    For each tile
      If (tile center vector is close to aperture vector)
        Add stars in tile to candidate star list
  
  \}

Method 2:
For each Aperture
  Form Bounding Box around final aperture of last period and final aperture of current period
  For each tile
    If (tile boundaries in bounding box)
      Add stars in tile to candidate star list
Determine Stars in FOV & CCD

For every update period
{
    Update FAME Attitude
    Update FAME Rate
    Compute Aperture vectors
    Get Candidate Stars
    Compute Tile Aberration
    For each candidate star
    {
        Correct for Parallax
        Correct for Stellar Aberration
        Project on to Focal Plane
        For each CCD
        {
            Compute Crossing time & position
        }
    }
}
Generate CCD Commands

For CCD
  For each star with crossing time
    {
      Form charge injection command
      If no window in keep out zone
        Add command to command queue
      For star window
        {
          Form CCD read commands
          Add commands to command queue
        }
      If previous charge injection keep out zone overlaps window
        Remove charge injection command
    }
Processing Requirement
Scalar Computations

- **CCD commands**
  - Estimated 100 OPS per command
  - 13 commands per window

- **Star Retrieval**
  - Estimated 100 OPS per star

- **Construction Guide Star Window**
  - 2 guide stars/sec
  - 200 pixels/star
  - 10 OPS/pixel
  - 4,000 OPS
# Floating Pt Computations

<table>
<thead>
<tr>
<th>Computation</th>
<th>Equation</th>
<th>Fmult</th>
<th>Fadds</th>
<th>Fsqrt</th>
<th>Fother</th>
<th>Fmult/s</th>
<th>Fadd/s</th>
<th>Fsqrt/s</th>
<th>Fother/s</th>
<th>Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update F</td>
<td>F + F'.dt</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Update x(t), y(t), z(t)</td>
<td>x(t) = x(t0) + W dt; y(t) = y(t0) + W dt; etc</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Update q</td>
<td>-x(t) sin(\phi/2) + y(t) cos(\phi/2)</td>
<td>6</td>
<td>3</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Total per update</td>
<td></td>
<td>24</td>
<td>18</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Select Tile</td>
<td>p.q &gt; 0.999759</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>18</td>
<td>12</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Compute tile aberration</td>
<td></td>
<td></td>
<td>B + F'/c</td>
<td></td>
<td></td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>Total (Tiles)</td>
<td></td>
<td>36</td>
<td>27</td>
<td>18</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Compute parallax</td>
<td>p - F</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Compute star aberration</td>
<td>p + A</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2,000</td>
<td>0,10%</td>
<td></td>
</tr>
<tr>
<td>Project star on focal plane</td>
<td>p.u, p.v</td>
<td>6</td>
<td>4</td>
<td>12,000</td>
<td>8,000</td>
<td>0</td>
<td>0</td>
<td>2,000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Determine pixel crossing time</td>
<td>t + u.(h-s)/w</td>
<td>4</td>
<td>6</td>
<td>8,000</td>
<td>12,000</td>
<td>0</td>
<td>0</td>
<td>2,000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total (Stars)</td>
<td></td>
<td>20,000</td>
<td>26,006</td>
<td>8,000</td>
<td>12,000</td>
<td>0</td>
<td>0</td>
<td>2,000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Grand Total</td>
<td></td>
<td>46,120</td>
<td>20,060</td>
<td>26,051</td>
<td>3,006</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend**

- **# dims**: Number of dimensions
- **Parameters**
  - Number stars/period: 2,000
  - % Parallax stars: 0.10%
  - Candidate tiles: 6
  - %Selected Tiles: 50%

- **Fame position vector**: F
- **Fame velocity vector**: F'
- **Delta time**: dt
- **Star position in (x,y,z) [ICRS]**: p
- **Spacecraft orientation**: x(t), y(t), z(t)
- **Aperture in (x,y,z) [ICRS]**: q
- **Velocity of light**: c
- **Aperture angle**: phi
- **Abberation**: A
- **Box center vector**: B
- **Instrument coordinates**: q, u, v
- **Spacecraft coordinates**: x, y, z
- **Star position in (u,v)**: p'
- **Pixel coordinates (u,v)**: h
- **Rotation velocity**: w
Processor Capability
CPU Performance

- **Integer operations**
  - Approximately one clock cycle per operation
- **Floating point operations**
  - Multiple cycles per operation (often 5-20)
  - Sometimes have add/multiply operation (i.e. faster dot products)
  - Older processors require more cycles for double precision
- **Dependent on memory organization**
  - Cache levels and size
- **Best test of performance is a prototype of the kernel code on an actual processor**
CPU Performance

- Assume 300% Margin
- Assume double precision
Star Processing Prototype
Activity

- Develop a prototype of all the algorithms used to perform star processing
  - C/C++ developed on PC
- Time performance on various processors
- Scale performance to selected processor
Status

- Window formation is nearly complete
  - Preliminary results at IPDR
- Catalog management under way
Summary

- Science requirement drives the selection and design of the software
- Estimated performance requires a 40+ MHz processor to provide margin
Backup
Use of Quad-Trees

Motion in 1 Second

FOV
Direction of Rotation
Bounding Box
Floating Pt Test Program

/* Floating Addition */
for (i=0; i<ARRAY_SIZE; i++)
    a[i]=b[i] = 1;
start=clock();
for (j=0; j<ITERATIONS; j++){
    for (i=0; i<ARRAY_SIZE; i++)
        a[i]=a[i]+b[i];
}
end=clock();
cpu_add = ((float)(end-start)/CLOCKS_PER_SEC);
adds = (ARRAY_SIZE*ITERATIONS)/cpu_add)/1e6;
cout << "Addition loop " << cpu_add << " CPU seconds" << endl;

/* Floating Multiplication */
for (i=0; i<ARRAY_SIZE; i++)
    a[i]=b[i] = 1;
start=clock();
for (j=0; j<ITERATIONS; j++){
    for (i=0; i<ARRAY_SIZE; i++)
        a[i]=b[i]*b[i];
}
end=clock();
cpu_mult = ((float)(end-start)/CLOCKS_PER_SEC);
mults = (ARRAY_SIZE*ITERATIONS)/cpu_mult)/1e6;
cout << "Multiplication loop " << cpu_mult << " CPU seconds" << endl;

/* Dot Product */
for (i=0; i<ARRAY_SIZE; i++)
    a[i]=b[i] = 1;
a_dot_b = 0;
start=clock();
for (j=0; j<ITERATIONS; j++){
    for (i=0; i<ARRAY_SIZE; i++)
        a[i]=b[i];
    a_dot_b += a[i]*b[i];
}
end=clock();
# Processor Performance (Mops)

<table>
<thead>
<tr>
<th></th>
<th>Fadd</th>
<th>Fmult</th>
<th>Fdot</th>
<th>DFadd</th>
<th>DFmult</th>
<th>DFdot</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerPC 603 75MHz</td>
<td>3.6</td>
<td>3.6</td>
<td>9.4</td>
<td>3.6</td>
<td>3.4</td>
<td>9.4</td>
</tr>
<tr>
<td>PowerPC 601 75MHz</td>
<td>4.6</td>
<td>4.6</td>
<td>10.6</td>
<td>4.7</td>
<td>4.7</td>
<td>10.7</td>
</tr>
<tr>
<td>PowerPC 601 100MHz</td>
<td>6.2</td>
<td>6.2</td>
<td>14.2</td>
<td>6.3</td>
<td>6.3</td>
<td>14.3</td>
</tr>
<tr>
<td>Pentium 233Mhz MMX</td>
<td>27.4</td>
<td>21.3</td>
<td>38.2</td>
<td>27.4</td>
<td>19.6</td>
<td>21.1</td>
</tr>
<tr>
<td>Pentium II 266Mhz</td>
<td>50.5</td>
<td>50.5</td>
<td>181.8</td>
<td>62.9</td>
<td>42.3</td>
<td>172.4</td>
</tr>
<tr>
<td>Pentium II 300MHz</td>
<td>58.7</td>
<td>58.7</td>
<td>148.9</td>
<td>72.3</td>
<td>48.5</td>
<td>196.85</td>
</tr>
</tbody>
</table>