Scalability of Table Lookup Techniques

Rick Langston, SAS Institute Inc.
Outline

- 5 different table lookup techniques shown
- Their performance is compared as we scale up
- Recommendations from that test scenario
- Macro for computing format memory usage
- More info on PROC FORMAT memory usage
Table Lookup Techniques

- Many different ways available
- Common question: which is fastest?
- We wanted to test scalability
- 1K, 10K, 100K, 500K, 1M, 5M, 10M obs
- 5 different methods chosen...
Using a format

/*-----PROC FORMAT and PUT function------*/
proc format cntlin=temp; run;
data _null_; set temp(rename=(label=shouldbe)) end=eof;
    label=put(start,$testfmt.);
    if label=shouldbe then matched+1;
    if eof;
    if matched=_n_ then put 'all matched';
    else put 'not all matched';
run;
Hash Object

/*-----hash object-----*/
data _null_; set temp(rename=(label=shouldbe)) end=eof;

    length label $20;
    retain label ' ';

    if _n_=1 then do;
        declare hash ht(datasets="temp");
        ht.defineKey("start");
        ht.defineData("label");
        ht.defineDone();
        end;

    rc = ht.find();
    if rc = 0 then do;
        if label=shouldbe then matched+1;
        end;
    if eof;
    if matched=_n_ then put 'all matched';
    else put 'not all matched';
run;
DATA step merge

/*----merge-----*/
proc sort data=temp out=temp2(drop=random rename=(label=shouldbe)); by start;
run;
proc sort data=temp out=lookup(drop=random); by start;
run;
data _null_; merge temp2(in.want) lookup end=eof; by start;
   if label=shouldbe then matched+1;
   if eof;
   if matched=_n_ then put 'all matched';
   else put 'not all matched';
run;
KEY= option for lookup

```/-----key= usage-----*/
data lookup(index=(start)); set temp(keep=start label);	run;
data _null_; set temp(keep=start label rename=(label=shouldbe)) end=eof;	set lookup key=start;	if label=shouldbe then matched+1;	if eof;	if matched=_n_ then put 'all matched';
else put 'not all matched';
run;```
SQL inner join

```sql
PROC SQL;
CREATE TABLE MERGED AS
    SELECT LABEL, SHOULDBE FROM TEMP A INNER JOIN TEMP(RENAME=(LABEL=SHOULDBE)) B
    ON A.START = B.START;
QUIT;

DATA _NULL_; SET MERGED END=EOF;
    IF LABEL=SHOULDBE THEN MATCHED+1;
    IF EOF;
    IF MATCHED=_N_ THEN PUT 'ALL MATCHED';
    ELSE PUT 'NOT ALL MATCHED';
RUN;
```
Operating systems used

- 64-bit Unix system
- 64-bit Windows system with 16G
- 32-bit Windows system with 2G
On the Unix box

- Up to 500K obs – formats and hash better
- At 700K – formats ran out of memory
- At 1.9M – hash runs out of memory
- Others OK up to 10M
- KEY= the slowest
- SQL join the best
On both Windows boxes

- All completed up to 10M
- SQL join seemed to do best
- Let’s look at the numbers shall we?
<table>
<thead>
<tr>
<th></th>
<th>1K</th>
<th>10K</th>
<th>100K</th>
<th>500K</th>
<th>1M</th>
<th>5M</th>
<th>10M</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format execution</td>
<td>0.01</td>
<td>0.02</td>
<td>0.25</td>
<td>1.56</td>
<td>1.96</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hash usage</td>
<td>0.06</td>
<td>0.05</td>
<td>0.26</td>
<td>1.58</td>
<td>3.61</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MERGE merging</td>
<td>0.04</td>
<td>0.08</td>
<td>0.44</td>
<td>1.63</td>
<td>4.36</td>
<td>21.70</td>
<td>44.09</td>
</tr>
<tr>
<td>SQL table creation</td>
<td>0.06</td>
<td>0.05</td>
<td>0.21</td>
<td>1.81</td>
<td>3.88</td>
<td>21.26</td>
<td>45.45</td>
</tr>
<tr>
<td>Key lookup</td>
<td>0.01</td>
<td>0.13</td>
<td>1.46</td>
<td>10.56</td>
<td>21.96</td>
<td>129.57</td>
<td>266.95</td>
</tr>
<tr>
<td>Windows 64-bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format execution</td>
<td>0.00</td>
<td>0.01</td>
<td>0.25</td>
<td>1.79</td>
<td>4.73</td>
<td>30.45</td>
<td>72.90</td>
</tr>
<tr>
<td>Hash usage</td>
<td>0.35</td>
<td>0.03</td>
<td>0.28</td>
<td>2.15</td>
<td>5.03</td>
<td>36.34</td>
<td>84.76</td>
</tr>
<tr>
<td>MERGE merging</td>
<td>0.05</td>
<td>0.02</td>
<td>0.37</td>
<td>1.33</td>
<td>3.01</td>
<td>25.44</td>
<td>49.60</td>
</tr>
<tr>
<td>SQL table creation</td>
<td>0.73</td>
<td>0.01</td>
<td>0.15</td>
<td>0.92</td>
<td>1.90</td>
<td>13.64</td>
<td>32.01</td>
</tr>
<tr>
<td>Key lookup</td>
<td>0.01</td>
<td>0.10</td>
<td>1.46</td>
<td>8.92</td>
<td>18.14</td>
<td>98.60</td>
<td>209.01</td>
</tr>
<tr>
<td>Windows 32-bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format execution</td>
<td>0.01</td>
<td>0.01</td>
<td>0.21</td>
<td>1.68</td>
<td>4.82</td>
<td>27.39</td>
<td>62.84</td>
</tr>
<tr>
<td>Hash usage</td>
<td>1.20</td>
<td>0.01</td>
<td>0.26</td>
<td>2.42</td>
<td>4.90</td>
<td>35.68</td>
<td>82.59</td>
</tr>
<tr>
<td>MERGE merging</td>
<td>0.10</td>
<td>0.03</td>
<td>0.20</td>
<td>1.47</td>
<td>3.33</td>
<td>16.66</td>
<td>27.49</td>
</tr>
<tr>
<td>SQL table creation</td>
<td>0.59</td>
<td>0.01</td>
<td>0.12</td>
<td>0.76</td>
<td>1.68</td>
<td>12.25</td>
<td>27.60</td>
</tr>
<tr>
<td>Key lookup</td>
<td>0.07</td>
<td>0.09</td>
<td>1.37</td>
<td>8.14</td>
<td>16.75</td>
<td>93.17</td>
<td>189.50</td>
</tr>
</tbody>
</table>
Memory constraints

- Windows 32 bit, Linux 32 bit, z/OS
- All at a max of 4G due to architecture
- 2G on Windows and z/OS
- 64-bit still constrained due to ....
- hardware memory
- SAS option choices
- adminstration
z/OS Constraints

- 32-bit system
- typically 64M is the max
- chargebacks convert memory to EXCPs
SAS options

- proc options group=memory define; run;
PROC FORMAT memory usage

- You want to create a set of large formats
- Or you already have these formats
- You need to know how much memory they will need
- Can they be simultaneously loaded?
Macro for computation

- %get_fmt_memsizes(mylib.formats);
- Creates an output data set with the memory needs for each member
- Available via http://support.sas.com
For formats not yet created

- Suppose you have 500,000 ranges/labels
- Create with the first 100 ranges/labels
- Run the macro
- Multiply the memory needed by 5000
Remember creation vs. usage

- More memory is needed to create a format than to use it
- But every associated format is loaded in the DATA or PROC step
- This applies regardless of explicit usage