Fair Tree

Fairshare Algorithm for Slurm

Ryan Cox and Levi Morrison
Brigham Young University
Outline

• Introduction to job prioritization
• BYU's setup
• Issues with existing fairshare algorithms
• Fair Tree
• Fairshare=parent on accounts
• Appendix
Job Prioritization

- Job Priority can include different components:
  - Age
  - JobSize
  - Partition
  - QOS
  - **Fairshare** - this is what we will discuss

- Job Priority = sum(ComponentWeight * ComponentFactor)
  - Weight is an integer
  - Factor is between 0.0 and 1.0

- Job Priority = 1000 * 0.34 + 5000 * 0.12 + 2000 * 0.54 ......
Fairshare Factor

- Usage and Shares are the two components of Fairshare Factor
- Raw Shares are assigned to each association by an admin
  - “Shares” is Raw Shares when normalized to 0.0 .. 1.0
  - Similar to slices of a pie
  - Represents the part of the system that is “yours”
- “Usage” is a value between 0.0 and 1.0 that represents your proportional usage of the system
Basic premise of fairshare:

If Shares == Usage, you have hit your “fairshare target”
Fairshare Equation

Fairshare Factor = $2^{-\text{Usage/Shares}}$

or

$F = 2^{-\text{U/S}}$

- Result is between 0.0 and 1.0
- When Usage increases, $F$ decreases
- When Usage decreases, $F$ increases
BYU's Setup

- Each professor has a Slurm account
  - Has *account coordinator* status for that account
  - BYU doesn't charge $$$ for usage
  - Accounts are treated equally (Shares are the same)
- Students and collaborators are users in a professor's account
  - Users are treated equally (Shares are the same, unless account coordinator changes them)
Our Definition of “Fair”

• If accounts A and B are siblings and A has a higher fairshare factor than B, all children of A will have higher fairshare factors than all children of B
  – True for all sibling associations at all levels
Disclaimer

• We will discuss BYU's use case and how to support it
  – It may also apply to many other use cases
• We are building on the hard work of others
• We identified some issues with existing algorithms but our work would not have been possible without the other algorithms
New Fairshare Algorithm

We wrote a new fairshare algorithm.

Why?
Account with 35% usage had higher priority than accounts with only 10% usage
Fairshare Equation

Fairshare Factor = $2^{\text{Usage/Shares}}$

or

$F = 2^{-\frac{U}{S}}$

The following slides show graphs of this equation with various $U$ and $S$ values.
Fairshare 3D Graph

\[ 2^{-(\text{Usage/Shares})} \]
Fairshare Graph

$2^-(\text{Usage/Shares})$

- Red line: Shares = 0.05 (20 Accounts)
- Blue line: Shares = 0.01 (100 Accounts)
- Pink line: Shares = 0.005 (200 Accounts)

Usage

Fairshare
As Usage increases, fairshare decreases.

As Shares decreases, the point at which the line visually hits the x-axis is earlier and earlier.

At some point it will also become zero in floating point.
Actual Fairshare Graph

$2^{\frac{1}{150}}$ (Usage/Shares)

- $1/150/10$ Shares ($0.00066666666667$)
- $1/150/50$ Shares ($0.00013333333333$)
Actual Fairshare Zoomed
Actual Fairshare More Zoomed

- The Shares values were seen in production at BYU
- 150 accounts with either 10 or 50 users in an account
- Even if the Shares value is the same, the Fairshare value may be the same (zero) even with a large disparity in Usage
- Since F was zero beyond a certain point, large disparities in usage resulted in same F value
Dampening Factor

- Added in 14.03
- \( 2^{-((\text{Usage}/\text{Shares})/\text{Damp})} \)
  - Damp is a positive integer (slurm.conf)
- Great idea
- Isn't a complete solution
  - Small values may still be too small
  - Requires manual guess and check
- Can be replaced by linear interpolation
  - Ticket-Based can use this instead of MAX()
  - See Appendix
Problem With Shares Calculation

- Normalized Shares equation (the “S” in $2^-(U/S)$) for all current algorithms:

\[ S = (\frac{\text{RawShares}_{\text{user}}}{\text{RawShares}_{\text{siblings}}}) \times 1/4 \times \left( \frac{\text{RawShares}_{\text{account}}}{\text{RawShares}_{\text{siblings-accounts}}} \right) \times 1/10 \times \left( \frac{\text{RawShares}_{\text{parent}}}{\text{RawShares}_{\text{parent-siblings}}} \right) \times \ldots \times 1/20 \times \ldots \]

\[ 1/4 \times 1/10 \times 1/20 = 1/800 = 0.00125 \]

- Assume sibling associations have the same Raw Shares, tree has 2 levels:

  Equivalent to:

  \[ S = \left( \frac{1}{\text{number of users in account}} \right) \times \text{constant} \]

- See appendix for demonstration of equivalence
Problem With Shares Calculation

The S values are multiplied by the same constant, thus the constant can be discarded for comparison purposes.

The S value ensures that AcctA has a huge advantage solely because it has fewer users.

End-users should attempt to get team members fired or preferably reassigned to a competing research group so that their own S increases while their competitors' S values decrease.

Problem applies to more complicated scenarios but is harder to model.

\[
\begin{align*}
S &= 1/2 = 0.5 \\
S &= 1/10 = 0.1
\end{align*}
\]
Usage Effective

- Usage Effective (the U in $2^-(U/S)$) =

  \[ UA_{child} + ((UE_{parent} - UA_{child}) \times S_{child}/S_{all_siblings}) \]

- Assuming two levels in the tree, $2^-(U/S)$ expands to:

  \[
  2 \left( \frac{U_{User}}{U_{Total}} + \left[ \frac{U_{Acct}}{U_{Total}} + \left( 1 - \frac{U_{Acct}}{U_{Total}} \right) \times \frac{S_{Acct}}{S_{Acct} + S_{ASiblings}} - \frac{U_{User}}{U_{Total}} \right] \times \frac{S_{User}}{S_{User} + S_{USiblings}} \right)
  \]

- Has some unpleasant side effects that we consider bugs
  - (Have 20 minutes and a whiteboard? Let's talk!)
Other Algorithms

- **Ticket-Based**
  - Depends on queue state
    - Hard to explain to users
  - \( T = T_{\text{parent}} \times S \times F / \text{SUM}(S\times F)_{\text{active_siblings}} \)
    - Similar issues to Norm Shares equation because \((S \times F / \text{SUM}(S\times F)_{\text{active_siblings}})\) depends on the number of active users in an account

- **Depth Oblivious**
  - Very complex math
    - Hard to explain to users and admins
    - Hard to evaluate its fairness
  - Many opportunities for floating point precision issues
An Example
The Beatles and Elvis split the cost of a spaceship and crew.

One passenger spaceship.

Elvis pays 50%.

Beatles pay 50%.

- Each member pays 25% of the Beatles' 50% (equals 1/8 of the total).
- If a band member isn't waiting to use the spaceship, other band members can take a ride.
How well does Slurm perform in this scenario?
# What actually happens?

Simulated using the following account hierarchy:

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Norm Shares</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>0.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>0.500000</td>
</tr>
<tr>
<td>beatles</td>
<td>harrison</td>
<td>0.125000</td>
</tr>
<tr>
<td>beatles</td>
<td>lennon</td>
<td>0.125000</td>
</tr>
<tr>
<td>beatles</td>
<td>mccartney</td>
<td>0.125000</td>
</tr>
<tr>
<td>beatles</td>
<td>starr</td>
<td>0.125000</td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>0.500000</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>0.500000</td>
</tr>
</tbody>
</table>
Simulation Rules

- Elvis always has work queued
- \( n \) Beatles always have worked queued
- The other Beatles do not queue any jobs
- Only one core available
- Usage and fairshare calculations run in between each job
- The following pie charts show the percentage of the system that each account used
Traditional Multifactor

- 4 Beatles vs Elvis
  - Beatles: 0.36
  - Elvis: 0.64

- 3 Beatles vs Elvis
  - Beatles: 0.33
  - Elvis: 0.67

- 2 Beatles vs Elvis
  - Beatles: 0.29
  - Elvis: 0.71

- 1 Beatle vs Elvis
  - Beatles: 0.2
  - Elvis: 0.8
Ticket-Based

4 Beatles vs Elvis

3 Beatles vs Elvis

2 Beatles vs Elvis

1 Beatle vs Elvis
Depth Oblivious

4 Beatles vs Elvis

3 Beatles vs Elvis

2 Beatles vs Elvis

1 Beatle vs Elvis
Fair Tree

4 Beatles vs Elvis

3 Beatles vs Elvis

2 Beatles vs Elvis

1 Beatle vs Elvis
Goals of Fair Tree

- If accounts A and B are siblings and A has a higher fairshare factor than B, all children of A will have higher fairshare factors than all children of B
  - True for all sibling associations at all levels
- Eliminate problems due to floating point precision loss
- sshare and debug support
Trees

- Similar idea to binary search tree:
  - Nodes are sorted
  - In-order traversal results in a sorted list
Fairshare Calculation

- Level Fairshare = S / U
  - $S = \frac{\text{RawShares}_{\text{self}}}{\text{RawShares}_{\text{self}+\text{siblings}}}$
  - $U = \frac{\text{RawUsage}_{\text{self}}}{\text{RawUsage}_{\text{self}+\text{siblings}}}$
- Range: $[0, \infty]$ 
  - if $U == S$ then $LF = 1.0$
- Replaces $2^{-\left(U/S\right)}$
- Only used for sorting
Tree Traversal

- Create “rooted plane tree”
- Traversal function, starting at root:
  - Calculate Level Fairshare for each child (S/U)
  - Sort children by Level Fairshare from highest to lowest
  - Visit the children in order
    - If account, recurse with account as new root
    - If user, assign a final fairshare factor based on ranking
- Final fairshare factor is: rank-- / user_count
  - rank is initialized to user_count
  - Ties are allowed
Traversal in the Fair Tree Algorithm for Slurm
Fair Tree: Traversal

- visit association
- calculate level fairshare
- sort siblings
- calculate final fairshare for user

Important Notes: Example level fairshare values are completely unrealistic. Final fairshare value not shown.
Fair Tree: Traversal

```
root
  /       \
A1  A2     \\
  |      /  \
u11  u21  u22  \
    /       \\
   /         \
  u12  u23   \\
     /       \\
    /         \
   u231  u232  u233
```
Visit association

Fair Tree: Traversal
Fair Tree: Traversal

Calculate level fairshare
Fair Tree: Traversal

Calculate level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Visit association
Calculate level fairshare
Fair Tree: Traversal

Calculate level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Visit association
Final fairshare value = rank-- / user_count

1.0 = 7 / 7
Fair Tree: Traversal

Visit association
Fair Tree: Traversal

Calculate level fairshare
Fair Tree: Traversal

Calculate level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Visit association
Final fairshare value = rank-- / user_count

0.857 = 6 / 7
Fair Tree: Traversal

Visit association
Final fairshare value = rank-- / user_count

0.714 = 5 / 7
Fair Tree: Traversal

Visit association
Fair Tree: Traversal

\[
\text{Final fairshare value} = \frac{\text{rank--}}{\text{user\_count}}
\]

\[
0.571 = \frac{4}{7}
\]
Fair Tree: Traversal

Visit association
Fair Tree: Traversal

Final fairshare value = rank-- / user_count

0.429 = 3 / 7
Fair Tree: Traversal

Visit association
Fair Tree: Traversal

Calculate level fairshare
Calculate level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Sort by level fairshare
Fair Tree: Traversal

Visit association
Fair Tree: Traversal

Final fairshare value = rank-- / user_count

0.286 = 2 / 7
Fair Tree: Traversal

Visit association
Final fairshare value = rank-- / user_count
Fair Tree: Traversal Complete

Traversal Complete!
All final fairshare values were assigned
Ranking

• Users are ranked as they are found
  – Ties are allowed (e.g. 8,8,8,5,4) if Level Fairshare is equal
  – See the appendix for details of tie handling
• Avoids precision loss
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[
\text{assoc} / (\text{assoc} + \text{siblings})
\]

**Level FS** (*with -l option*) – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

```
<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>0.909763</td>
<td>0.561462</td>
</tr>
<tr>
<td>beatles</td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.445266</td>
<td>0.200000</td>
<td>0.561462</td>
</tr>
<tr>
<td>beatles</td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.150888</td>
<td>0.600000</td>
<td>1.656863</td>
</tr>
<tr>
<td>beatles</td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.054734</td>
<td>0.800000</td>
<td>4.567568</td>
</tr>
<tr>
<td>beatles</td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.400000</td>
<td>0.716102</td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.110108</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
</tbody>
</table>
```

*Note: Some sshare columns were cut for brevity in this presentation*
slurmctl: Fair Tree fairshare algorithm, starting at root:
slurmctl:     elvis (elvis):  1.11010830324909747294
slurmctl:     elvis (elvis):  1.00000000000000000000
slurmctl:     beatles (beatles):  0.90976331360946745562
slurmctl:         mccartney (beatles):  4.56756756756756756785
slurmctl:         lennon (beatles):  1.65686274509803921568
slurmctl:         starr (beatles):  0.71610169491525423724
slurmctl:         harrison (beatles):  0.56146179401993355479

- Live view of depth-first traversal
- Users are printed in order of final fairshare factor
- Number shown is Level FS
Easy to Try

- slurm.conf:
  - PriorityFlags=FAIR_TREE
- scontrol reconfigure
  - Wait for next iteration ($PriorityCalcPeriod minutes)
- Available starting in 14.11.0pre6
Advantages

- Uses entry-level computer science (ordered tree)
- Uses entry-level math
  - S/U is simple
  - LevelFS < LevelFS is simple
- Precision loss issues extremely unlikely
- Handles unbalanced trees
- Demonstrably fair
Possible Concerns

• A user can use all the account's shares if no one else is running
  - (Great!)
  - No algorithm can prevent this
  - User limits (set by admin or account coordinator)

• Tiny user in a very active group has trouble running
  - User will have highest fairshare factor in account but low compared to other accounts
  - Optional QOS w/UsageFactor=20 and Priority=100000000
    • User runs quickly but pays heavily
  - Secondary account with tight limits

• Strictness of hierarchical prioritization
  - Fairshare=parent on accounts
  - Fuzzy matching (see “Future Development”)

Fairshare=parent modifications (all algorithms)
Fairshare=parent

- Fairshare=parent behavior on an account was previously undefined
- It is now defined:

If Fairshare=parent is set on an account, that account's children will be effectively reparented for fairshare calculations to the first ancestor that is not Fairshare=parent.

- The behavior of limits is unchanged
- Available in 14.11 for all algorithms
Fairshare=parent
Fairshare=parent

- root
  - Engineering
    - ElecAndComp
      - ElecE
        - DrEE1
          - u1
          - u2
      - CompE
        - DrCE1
          - u3
          - u4
        - DrCE2
          - u5
          - u6
          - u7
    - ME
      - DrME1
        - u8
        - u9
      - DrME2
        - ua
        - ub
  - PhysMath
    - Physics
      - DrPhys1
        - uc
        - ud
        - ue
      - DrPhys2
        - uf
        - ug
        - uh
    - CS
      - DrCS1
        - ui
        - uj
        - uk
      - DrCS2
        - ul
        - um
    - Chemistry
      - DrChem1
      - DrChem2
Fairshare=parent
Fairshare=parent

```
root
|
A1
u11  u12
|
A2
u21  ACollab
|
   |
   u221  u222
   |
   |
   u231  u232  u233
```


Fairshare=parent

Example:

ACollab is the only association with Fairshare=parent
Example:

ACollab effectively disappears
Fairshare=parent

Example:
\textit{u221} and \textit{u222} are reparented
The following are now **siblings** for purposes of fairshare calculations:
u21, u221, u222, A23
The following are now **siblings** for purposes of fairshare calculations:

u21, u221, u222, A23
Questions?
Appendix Index

- More information
- Future Development
- Problems with Shares Calculation
- Linear Interpolation
- sshare example
- Tie Handling
More Information

• Until 14.11 is released, Fair Tree documentation is at:
  – https://fsl.byu.edu/documentation/slurm/fair_tree.php
• Available in 14.11.0pre6
• Decay is orthogonal to fairshare, i.e. it still works fine
• Contact us if you have questions:
  – ryan_cox@byu.edu
  – levi_morrison@byu.edu
  – slurm-dev list
Future Development

- Allow for less strictness
  - Merge accounts based on configurable delta
    - Allows reuse of tie handling. The comparison function needs a very minor change
  - LevelFSTieDelta=0.01,0.05,0.02
    - Values are the epsilon value at increasing depths
  - Either:
    - $a > b \times (1 - \delta)$ && $a < b \times (1 + \delta)$
    - $\text{abs}((a - b) < \delta)$
Problems With Shares Calculation

• Normalized Shares equation for all current algorithms:

\[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \left( \frac{S_{\text{account}}}{S_{\text{Sibling-accounts}}} \right) \times \left( \frac{S_{\text{parent}}}{S_{\text{parent-siblings}}} \right) \times \ldots \]
Problems With Shares Calculation

- Normalized Shares equation for all current algorithms:

\[ S = \left( \frac{S_{user}}{S_{siblings}} \right) \times \left( \frac{S_{account}}{S_{sibling-accounts}} \right) \times \left( \frac{S_{parent}}{S_{parent-siblings}} \right) \times \ldots \]

Example:
- The tree is no deeper than: root->account->user
- Assume sibling associations are treated equally (same Shares)
- Problem still applies for more complicated scenarios
Problems With Shares Calculation

• Normalized Shares equation for all current algorithms:

\[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \left( \frac{S_{\text{account}}}{S_{\text{sibling-accounts}}} \right) \times \left( \frac{S_{\text{parent}}}{S_{\text{parent-siblings}}} \right) \times \ldots \]

• The tree is no deeper than: root->account->user

• Therefore:

\[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \left( \frac{S_{\text{account}}}{S_{\text{sibling-accounts}}} \right) \]
Problems With Shares Calculation

• Normalized Shares equation for all current algorithms:

\[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \left( \frac{S_{\text{account}}}{S_{\text{sibling-accounts}}} \right) \times \left( \frac{S_{\text{parent}}}{S_{\text{parent-siblings}}} \right) \times \ldots \]

• If each account is Shares=500

\[ \frac{S_{\text{account}}}{S_{\text{sibling-accounts}}} = \frac{500}{(500 \times \text{count(accounts)})} \]

\[ \frac{S_{\text{account}}}{S_{\text{sibling-accounts}}} = \frac{1}{\text{count(accounts)}} \]

• This is a constant for all accounts
Problems With Shares Calculation

- Normalized Shares equation for all current algorithms:
  \[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \text{constant} \]
  \[ \times \left( \frac{S_{\text{account}}}{S_{\text{siblings}}} \right) \]
  \[ \times \left( \frac{S_{\text{parent}}}{S_{\text{siblings}}} \right) \times \ldots \]

- \[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \times \text{constant} \]

- Since we are comparing... constants are thrown out:
  \[ S = \left( \frac{S_{\text{user}}}{S_{\text{siblings}}} \right) \]
Problems With Shares Calculation

- Normalized Shares equation for all current algorithms:

\[ S = \left( \frac{S_{user}}{S_{siblings}} \right) \times \text{constant} \]
\[ \times \left( \frac{S_{account}}{S_{sibling-accounts}} \right) \]
\[ \times \left( \frac{S_{parent}}{S_{parent-siblings}} \right) \times \ldots \]

- If each user is Shares=100
- \[ S = \frac{100}{(100 \times \text{count(users_in_account))}} \]
- \[ S = \frac{1}{\text{count(users_in_account)}} \]
Problems With Shares Calculation

- Normalized Shares equation for all current algorithms when accounts are treated equally and users in an account are treated equally:

\[ S_{\text{user}} = \frac{1}{\text{count(users\_in\_account)}} \times \text{constant} \]

The problem still exists for other scenarios but is harder to model.
Problems With Shares Calculation

\[ S = \frac{1}{2} = 0.5 \]

\[ S = \frac{1}{10} = 0.1 \]
Linear Interpolation (lerp)

- Shares are between 0.0 .. 1.0
- Map them to the range 0.1 .. 1.0
  - Use linear interpolation (lerp)
- \[ \text{lerp}(\text{min}, \text{max}, f) = \text{min} + f \times (\text{max} - \text{min}) \]
  - Has floating point issues
- \[ \text{lerp}(\text{min}, \text{max}, f) = \text{min} \times (1.0L - f) + \text{max} \times f \]
  - Shouldn't have floating point issues
- \[ \text{lerp}(0.1, 1.0, 0.00001) = 0.100009 \]
Linear Interpolation (lerp)

\[ 2^{-(\text{Usage/Shares})} \]

- Shares: 1/50/50 = 0.000133333333333
- Shares: lerp(1/50/150) = 0.1006
## Unmodified vs lerp()

<table>
<thead>
<tr>
<th>Usage</th>
<th>Shares</th>
<th>Fairshare (unmodified)</th>
<th>Fairshare (lerp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15</td>
<td>.2</td>
<td>0.59460355750136062447097629046766</td>
<td>0.68981706017270247755391460176000</td>
</tr>
<tr>
<td>0.15</td>
<td>.1</td>
<td>0.35355339059327384187980669594253</td>
<td>0.57855511864135993283170603107557</td>
</tr>
<tr>
<td>0.15</td>
<td>.02</td>
<td>0.00552427172801990291201024163570</td>
<td>0.41431890600793676033685580817334</td>
</tr>
<tr>
<td>0.15</td>
<td>.01</td>
<td>0.00030517578125000000000000000000</td>
<td>0.38524635476490996013063453085046</td>
</tr>
<tr>
<td>0.15</td>
<td>.002</td>
<td>0.000000000000000000000002646977960</td>
<td>0.36011325130027945195512403819826</td>
</tr>
<tr>
<td>0.15</td>
<td>.001</td>
<td>0.00000000000000000000000000000000</td>
<td>0.35684750329912502426078085848715</td>
</tr>
<tr>
<td>0.15</td>
<td>.0002</td>
<td>0.00000000000000000000000000000000</td>
<td>0.35421449328863424725827649397480</td>
</tr>
<tr>
<td>0.15</td>
<td>.0001</td>
<td>0.00000000000000000000000000000000</td>
<td>0.35388408479113951296806533930450</td>
</tr>
<tr>
<td>0.15</td>
<td>.00002</td>
<td>0.00000000000000000000000000000000</td>
<td>0.35355339065944120049999645216676</td>
</tr>
<tr>
<td>0.15</td>
<td>.000001</td>
<td>0.00000000000000000000000000000000</td>
<td>0.35355339062635748816239289471497</td>
</tr>
</tbody>
</table>

- Fairshare factor is multiplied by PriorityWeightFairshare, making the problem much worse
- If lerp() is implemented, add an option (-L?) to sshare:
  - change Norm Shares → Lerp Shares
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[ \text{assoc} / (\text{assoc}+\text{siblings}) \]

**Level FS** *(with -l option)* – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.445266</td>
<td>0.200000</td>
<td>0.909763</td>
</tr>
<tr>
<td>beatles</td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.150888</td>
<td>0.600000</td>
<td>1.656863</td>
</tr>
<tr>
<td>beatles</td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.054734</td>
<td>0.800000</td>
<td>4.567568</td>
</tr>
<tr>
<td>beatles</td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.400000</td>
<td>0.716102</td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.110108</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

**McCartney:** Why do I have a lower fairshare factor than Elvis?

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[ \text{assoc} / (\text{assoc} + \text{siblings}) \]

**Level FS** (*with -l option*) – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>0.909763</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.244715</td>
<td>0.561462</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.150888</td>
<td>1.656863</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.054734</td>
<td>4.567568</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.716102</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.110108</td>
<td></td>
</tr>
</tbody>
</table>

**McCartney:** Why do I have a lower fairshare factor than Elvis?

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effective Usage** – reflect the Fair Tree approach

\[
\text{assoc} / (\text{assoc}+\text{siblings})
\]

**Level FS** (*with -l option*) – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effective Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>0.909763</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.244715</td>
<td>0.561462</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.082927</td>
<td>0.600000</td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.030081</td>
<td>0.561462</td>
<td>0.800000</td>
</tr>
<tr>
<td>beatles</td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.191870</td>
<td>0.716102</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.110108</td>
<td></td>
</tr>
</tbody>
</table>

McCartney: Why do I have a lower fairshare factor than Elvis?

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[
\text{assoc} \div (\text{assoc} + \text{siblings})
\]

**Level FS** *(with -l option)* – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>1230</td>
<td>0.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>676</td>
<td>0.500000</td>
<td>0.549593</td>
<td>0.549593</td>
<td></td>
<td>0.909763</td>
<td></td>
</tr>
<tr>
<td></td>
<td>harrison</td>
<td>301</td>
<td>0.250000</td>
<td>0.445266</td>
<td>0.200000</td>
<td></td>
<td>0.561462</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lennon</td>
<td>102</td>
<td>0.250000</td>
<td>0.150888</td>
<td>0.600000</td>
<td></td>
<td>1.656863</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mccartney</td>
<td>37</td>
<td>0.250000</td>
<td>0.054734</td>
<td>0.800000</td>
<td></td>
<td>4.567568</td>
<td></td>
</tr>
<tr>
<td></td>
<td>starr</td>
<td>236</td>
<td>0.250000</td>
<td>0.349112</td>
<td>0.400000</td>
<td></td>
<td>0.716102</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>554</td>
<td>0.500000</td>
<td>0.450407</td>
<td>0.450407</td>
<td></td>
<td>1.110108</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>554</td>
<td>1.000000</td>
<td></td>
<td>1.000000</td>
<td></td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

**Answer:** Compare the accounts' Level FS values

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach 
\[ \text{assoc} / (\text{assoc} + \text{siblings}) \]

**Level FS** *(with -l option)* – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>0.909763</td>
<td></td>
</tr>
<tr>
<td></td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.244715</td>
<td>1.110108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.082927</td>
<td>0.600000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.030081</td>
<td>0.800000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.191870</td>
<td>0.400000</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td></td>
<td>1.110108</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td></td>
<td>1.000000</td>
</tr>
</tbody>
</table>

**Answer:** Compare the accounts' Level FS values

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[ \text{assoc} / (\text{assoc} + \text{siblings}) \]

**Level FS** (*with -l option*) – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td></td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td>0.909763</td>
<td></td>
</tr>
<tr>
<td></td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.244715</td>
<td>0.561462</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.150888</td>
<td>1.656863</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.054734</td>
<td>4.567568</td>
<td></td>
</tr>
<tr>
<td></td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.716102</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.110108</td>
<td></td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
</tr>
</tbody>
</table>

**McCartney:** How is Level FS calculated?

*Note: Some sshare columns were cut for brevity in this presentation*
• **Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach
  \[
  \text{assoc} / (\text{assoc} + \text{siblings})
  \]

• **Level FS** (*with* `-l` *option*) – Fairshare compared to siblings

• **FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>beatles</td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.445266</td>
<td>0.561462</td>
</tr>
<tr>
<td></td>
<td>beatles</td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.150888</td>
<td>1.656863</td>
</tr>
<tr>
<td></td>
<td>beatles</td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.054734</td>
<td>4.567568</td>
</tr>
<tr>
<td></td>
<td>beatles</td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.716102</td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.110108</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td>1.000000</td>
<td></td>
</tr>
</tbody>
</table>

**Answer:** Norm Shares / Effectv Usage

*Note: Some sshare columns were cut for brevity in this presentation*
**Norm Shares** and **Effectv Usage** – reflect the Fair Tree approach

\[
\text{assoc} / (\text{assoc}+\text{siblings})
\]

**Level FS** *(with -l option)* – Fairshare compared to siblings

**FairShare** – This is the final fairshare factor (i.e. it works like it should)

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>Raw Shares</th>
<th>Norm Shares</th>
<th>Raw Usage</th>
<th>Norm Usage</th>
<th>Effectv Usage</th>
<th>FairShare</th>
<th>Level FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td></td>
<td>0.000000</td>
<td>1230</td>
<td>1.000000</td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>beatles</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>676</td>
<td>0.549593</td>
<td>0.549593</td>
<td></td>
<td>0.909763</td>
</tr>
<tr>
<td></td>
<td>harrison</td>
<td>25</td>
<td>0.250000</td>
<td>301</td>
<td>0.244715</td>
<td>0.244715</td>
<td>0.445266</td>
<td>0.561462</td>
</tr>
<tr>
<td></td>
<td>lennon</td>
<td>25</td>
<td>0.250000</td>
<td>102</td>
<td>0.082927</td>
<td>0.082927</td>
<td>0.150888</td>
<td>1.656863</td>
</tr>
<tr>
<td></td>
<td>mccartney</td>
<td>25</td>
<td>0.250000</td>
<td>37</td>
<td>0.030081</td>
<td>0.030081</td>
<td>0.054734</td>
<td>4.567568</td>
</tr>
<tr>
<td></td>
<td>starr</td>
<td>25</td>
<td>0.250000</td>
<td>236</td>
<td>0.191870</td>
<td>0.191870</td>
<td>0.349112</td>
<td>0.716102</td>
</tr>
<tr>
<td>elvis</td>
<td></td>
<td>500</td>
<td>0.500000</td>
<td>554</td>
<td>0.450407</td>
<td>0.450407</td>
<td></td>
<td>1.110108</td>
</tr>
<tr>
<td>elvis</td>
<td>elvis</td>
<td>1</td>
<td>1.000000</td>
<td>554</td>
<td>0.450407</td>
<td>1.000000</td>
<td></td>
<td>1.000000</td>
</tr>
</tbody>
</table>

**Answer**: Norm Shares / Effectv Usage

*Note: Some sshare columns were cut for brevity in this presentation*
Ties

- Occur when sibling associations have the same level_fs
  - Extremely rare except when raw usage == 0.0
  - Added complexity to the code

- Rules:
  - Sibling **users** receive the same rank
  - Sibling **accounts** have their children lists merged
  - A **user** with the same level_fs as a sibling **account** will receive the same rank as the account's highest ranked user
This is one portion of the tree. AT is pictured in its entirety.
There are 83 users (not all pictured). Some have been visited.
There are 83 users (not all pictured). Some have been visited.

rank = 64
Fair Tree: Tie Handling

rank = 64
Fair Tree: Tie Handling

```
    root
   /   |
  AT   A1
 /     |
A2     u3
       /|
   u4   u2a|
         /|
        u2b u2c

rank = 64
```
rank = 64
Fair Tree: Tie Handling

rank = 64
Fair Tree: Tie Handling

rank = 64
Fair Tree: Tie Handling

u3, A1, and A2 are tied at .6

rank = 64
Fair Tree: Tie Handling

Tied users are placed to the left

\[ \text{rank} = 64 \]
Fair Tree: Tie Handling

rank = 64
Fair Tree: Tie Handling

\[
\text{root} \quad \text{AT} \\
\text{u3 (.6)} \quad \text{A1 (.6)} \quad \text{A2 (.6)} \\
\text{u4 (.8)} \\
\text{u1a} \quad \text{u1b} \\
\text{u2a} \quad \text{u2b} \quad \text{u2c} \\
\text{rank} = 64
\]
Fair Tree: Tie Handling

Next association not tied. Decrement rank.

rank = 63
Fair Tree: Tie Handling

rank = 63
Fair Tree: Tie Handling

Next association \textbf{is} tied. rank stays the same

rank = 63
Next association **is** tied. rank stays the same

```
rank = 63
```
Fair Tree: Tie Handling

Peek at next account to check for a tie

rank = 63
Fair Tree: Tie Handling

A1->level_fs == A2->level_fs

rank = 63
Fair Tree: Tie Handling

Accounts are equal. Merge!

rank = 63
Fair Tree: Tie Handling

Merge complete

rank = 63
Fair Tree: Tie Handling

Merge complete

rank = 63
Calculate level_fs using **actual siblings**, *not* adopted ones. The calculations pretend that the merge never happened.

\[ \text{rank} = 63 \]
Fair Tree: Tie Handling

Calculate level_fs using **actual siblings**, *not* adopted ones. The calculations pretend that the merge never happened.

`rank = 63`
Fair Tree: Tie Handling

Calculate level_fs using **actual siblings**, **not** adopted ones. The calculations pretend that the merge never happened.

rank = 63
Fair Tree: Tie Handling

Sorting uses the merged list

rank = 63
Fair Tree: Tie Handling

Sorting uses the merged list

rank = 63
Fair Tree: Tie Handling

Note that the merged users are mixed

rank = 63
Fair Tree: Tie Handling

rank = 63
Fair Tree: Tie Handling

Note that rank has not been decremented. u1b receives the same rank as u3 since A1 tied u3.

rank = 63
Next association is tied. Rank stays the same

rank = 63
Fair Tree: Tie Handling

\[ \text{root} \]

\[ \text{AT} \]

\[ \text{u4 (.8)} \quad \text{u3 (.6)} \quad \text{A1 (.6)} \]

\[ \begin{align*}
\text{u1b (1.0)} & \quad \text{u2b (1.0)} & \quad \text{u1a (.8)} & \quad \text{u2c (.4)} & \quad \text{u2a (.3)} \\
63 & \quad 63 & \quad 63 & \quad 63 & \quad 63 \\
\end{align*} \]

\[ \text{u3 == u1b == u2b} \]

\[ \text{rank = 63} \]
Next association not tied. Decrement rank by 3 (3 ties)

rank = 60
Fair Tree: Tie Handling

rank = 60
Fair Tree: Tie Handling

Next association not tied. Decrement rank

rank = 59
Fair Tree: Tie Handling

rank = 59
Fair Tree: Tie Handling

Next association **not** tied. Decrement rank

rank = 58
Fair Tree: Tie Handling

Sorting uses the merged list

rank = 58
Fair Tree: Tie Handling

u3, u1b, and u2b receive equal rankings
Fair Tree: Tie Handling

- u3 tied A1, thus it tied A1's highest rank user, u1b
- u1b tied its adopted sibling user, u2b, since A1 and A2 were tied
- u3, u1b, and u2b receive the same fairshare factor