USER EXPERIENCES WITH A CHAPEL IMPLEMENTATION OF UTS

Claudia Fohry
FG Programmiersprachen/-methodik
Universität Kassel

Jens Breitbart
Rechnertechnik und Rechnerorganisation / Parallelrechnerarchitektur
TU München

WHY?

- Is Chapel easy to use?
- Which constructs work out well?
- Run through the learning process in Chapel

OVERVIEW

- UTS - What's that again?
- Our task pool
- Our implementation
- Wish list
- Conclusion

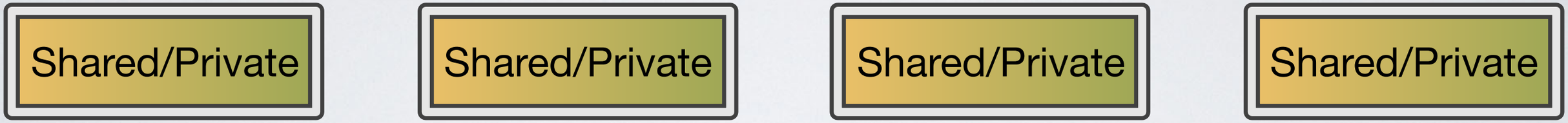
UTS

- Benchmark to study load balancing
- Extract nodes of a tree that is generated at runtime
- 1 task = extract all children of a node
- You'll need a task pool for that
  - and that was our focus :-)

TASK POOL

- One global pool for multiple locales and worker
- Stealing in a round robin fashion

TASK POOL



- One global pool for multiple locales and workers
- Stealing in a round robin fashion

OVERALL STRUCTURE

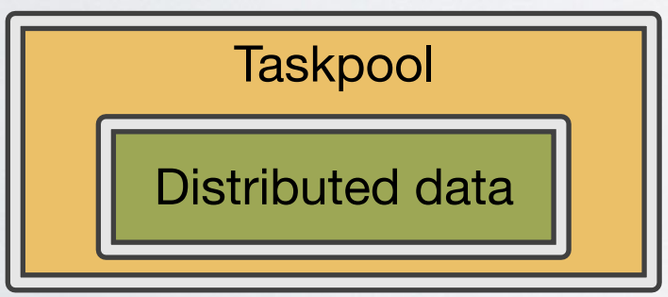
- The implementation consists of 5 modules
- UTS: configurable tree constants
- Node: TreeNode ~ one task (record or class?)
- Pool: our distributed data structure and counters
- Thread: worker "main" function

OBJECT ORIENTATION

- The code we want:

```
var pool :TaskPool;
```

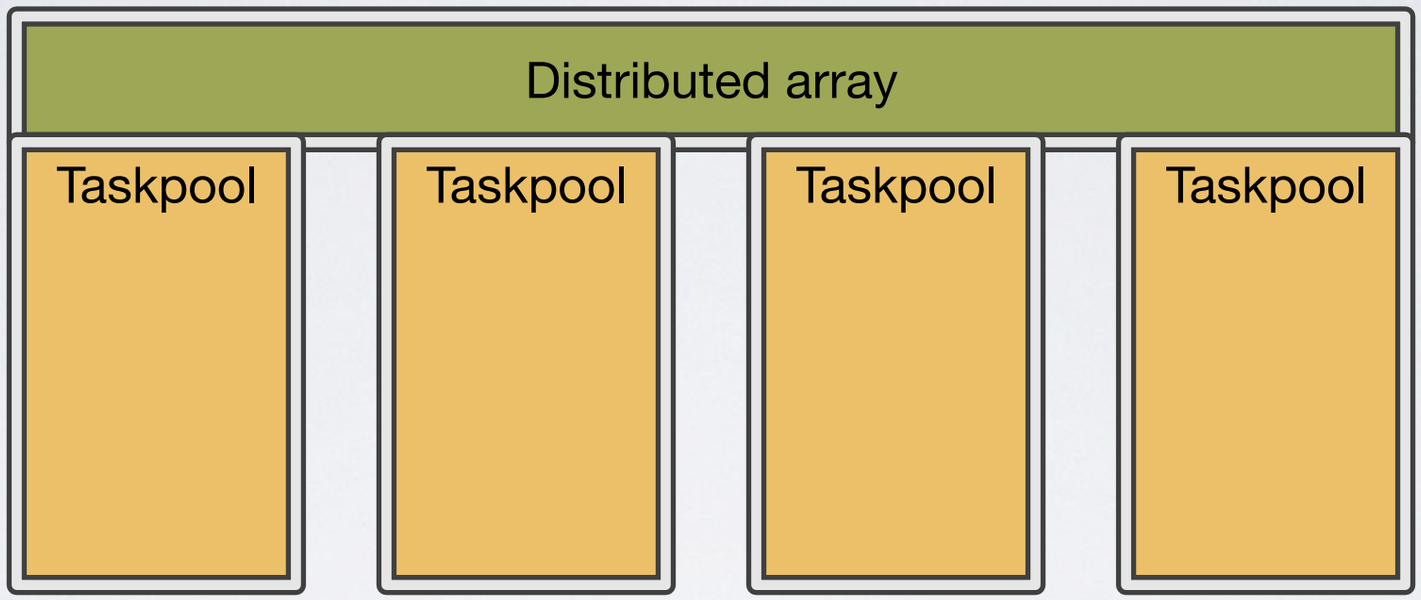
- The problem this code brings us:



The object has to live on one locale

... and we probably can't cache it.

DISTRIBUTE FIRST?



- Distribute first gives less encapsulation
- A place local handle would help, but not solve the problem

CLASS OR RECORD?

- Treenode as a record
  - Pool one chunk of memory per locale
  - Passing as a parameter involves copy
- Treenode as a class
  - Includes an indirection

SYNCHRONIZATION

- The number of publicly available tasks is stored as a synchronization variable

- We use it to rebuild a critical section

REDUCTION

- We need to reduce the final results
- Reduce requires a iterable data structure
- We implemented our own reduction
- I'd like a reduction variable, please

WISH LIST

- Local keyword for variables
- Predefined iterators ... maybe even one based on location of the data?
- on (<set of locales>)
- replicated distribution on scalars

CONCLUSION

- Well, I was picky ... but it worked out well and was rather easy

- Source code is available at: https://www.uni-kassel.de/eecs/fileadmin/datas/fb16/Fachgebiete/PLM/Dokumente/Publications_Fohry/ppam13uts.tar.gz