APGAS Programming in X10

Vijay Saraswat
Goldman Sachs

July 9, 2018

1 Resilient X10

1.1 Introduction

What happens if when working with many nodes, one of those nodes goes down?
APGAS: Asynchronous Partitioned Global Address Space
Resilient APGAS
HBI: Happens Before Invariance (Principle)

1.2 X10

- A language
  - Scala-like syntax
  - object-oriented, imperative, strongly typed, garbage collected
  - focus on scale → focus on parallelism and distribution
  - focus on productivity
- An implementation of the APGAS programming model
  - Asynchronous partitioned global access space
    - PGAS: Single address space but with internal structure (→ locality control)
    - asynchronous: task-based parallelism, active-message-based distribution
- A tool chain
  - compiler, runtime, standard library, IDE
  - started as open-source *research prototype, currently used for production*

PGAS: Consider a series of nodes each running a part of the program, each with different resources. The programmer has to worry about where the objects in the program reside.
SPMD (Single Program - Multiple Data): Single Program written in one place distributed across many nodes, then each node works until it hits a stopping point, then the code synchronizes. SPMD should run in APGAS efficiently.

1.2.1 Introducing Asynchrony, Designing APGAS

Message Passing - Each task lives in its own space (MPI)
Shared Memory - shared address space for all the tasks (OpenMP)
Task Parallelism:
- `async S`
- `finish S`

Place-Shifting Operations:
- `at(p) S`
- `at(p) e`

Concurrency control within a place:
- `when(c) S`
- `atomic S`

Distributed Heap:
- `GlobalRef[T]`
- `PlaceLocalHandle[T]`

`S` := Statement
`e` := Expression - Returns a Value
`p` := Place
`c` := Boolean Expression
`[T]` := Type
Atomic:
`S` is excecuted as if in a single step. Cannot contain `async, at, when`.