Adaptive Middleware and High Performance Software for Multi-core Deployments Across Cloud Configurations

R User Conference 2010

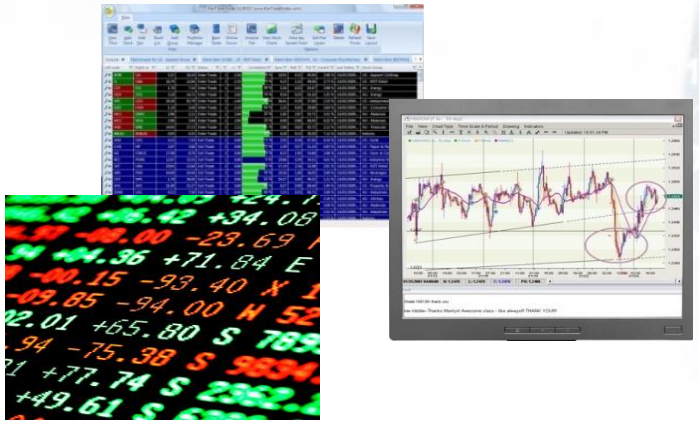
Douglas C. Schmidt
Chief Technology Officer
Zircon Computing

July 22, 2010

Motivation:
Accelerate Garrett Asset Management Models

Objective

- Run 100,000 Garrett Asset Management (GAM) R-based models 2 orders of magnitude faster than previous processing times



Motivation:
Accelerate Garrett Asset Management Models

Objective

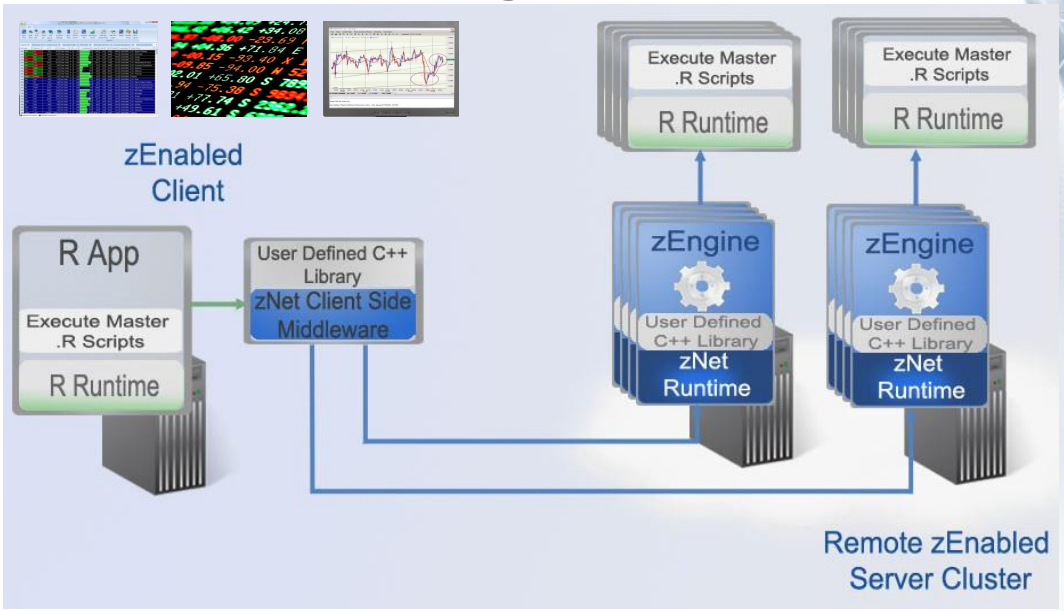
- Run 100,000 Garrett Asset Management (GAM) R-based models 2 orders of magnitude faster than previous processing times

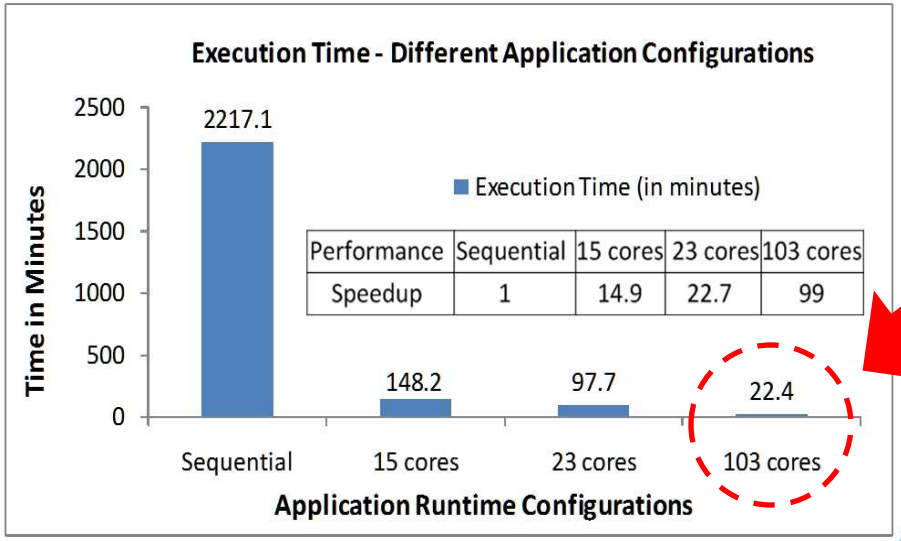
Approach

- Integrated R with Zircon Computing's adaptive high performance middleware

Results

- Reduced processing time from 2,217 minutes (36 hours) to 22 minutes (2 orders of magnitude) on commodity ~100 core cloud
- Solution was easy program, deploy, maintain, and administer





Overview of GAM Backtesting System

- The Garrett Asset Management (GAM) Backtesting System financial application guides future trading decisions by analyzing historical data to determine how a trading method would have performed in past stock markets
  - Executes a large # of logically independent and computationally intensive calculations to simulate behavior of models on historical data
  - For each combination of a model and a time period, the GAM Backtesting System performs computationally intensive calculation and collects results until computations are done

- R was natural choice to meet GAM's statistical analysis needs since its convenient and powerful abstractions allow users to run complex data analysis with relatively few commands

```
for (x in MODELS)
  for (I in i:length (StratPars)
    allres <- rbind (allres,
                     GenericgetNAVs (…))
```

The faster the R-based GAM models run, the more valuable the results

Initial R-based GAM Backtesting System

```
for (x in MODELS) {
  for (I in i:length (StratPars)
    allres <-
      rbind (allres,
              GenericgetNAVs (…))
```

```
GenericgetNAVs (…) {
  // Many compute-intensive
  // calculations written
  // using R function calls
}
```

StratPars is an matrix that is iterated across each row

The `GenericgetNAVs()` function performs backtesting on each row of historical data

Pros

- R-based application was straightforward to develop and evolve
  - Several 1,000 lines of R code
- Key R capabilities (re)used included matrices, RSQL, R statistics package, fTrading package, and TTR

Cons

- Problem 1: Performance was poor (~36 hours per model) due to R interpreter overhead
  - Problem exacerbated for large # of iterations in typical StratPars matrices
- Problem 2: R is single-threaded, which makes it hard to exploit modern multicore processors

Initial R-based GAM Backtesting System

```
for (x in MODELS) {
   allres <-
      call_get_navs (StratPars)
...
call_get_navs (StratPars, ...) {
  // Use .Call (R capability) to
  // call external C++ function
  fun <- "call_generic_get_navs"
  val <- .Call (fun, StratPars,…)
}
```

```
// This function is written in C++.
// Any R code can call this
// function after loading library
// containing this function.
call_generic_get_navs (StratPars) {
    for (int i = 0;
         i < length (StratPars);
         ++i)
      CPP_GenericgetNAVs (...)
}
```

Partially addressed problem 1 by moving one R loop to C++ function

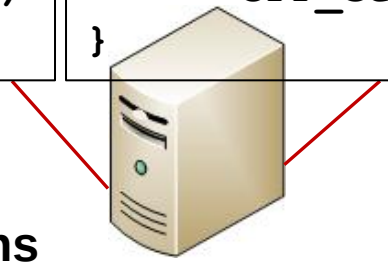
- Provide **`StratPars`** matrix as input to C++ function
- C++ function iterates through StratPars matrix and for each row calls **`CPP_GenericgetNAVs()`**

- Use Rcpp to send **`StratPars`** to C++ func **`call_generic_get_navs()`**
- Use RInside to call R function **`GenericgetNAVs()`** from C++ func **`CPP_GenericgetNAVs()`**

Initial R-based GAM Backtesting System

```
for (x in MODELS) {
   allres <-
      call_get_navs (StratPars)
...
call_get_navs (StratPars, ...) {
  // Use .Call (R capability) to
  // call external C++ function
  fun <- "call_generic_get_navs"
  val <- .Call (fun, StratPars,…)
}
```

```
// This function is written in C++.
// Any R code can call this
// function after loading library
// containing this function.
call_generic_get_navs (StratPars) {
    for (int i = 0;
         i < length (StratPars);
         ++i)
      CPP_GenericgetNAVs (...)
}
```



Pros

- By using Rcpp and RInside, we moved for loop execution from interpreted structure of R to compiled structure of C++, thus executing the inner loop faster

Cons

- Problem 2 still remains: loop iterations are sequential
- We therefore can't exploit remote/multicore processors to accelerate GAM Backtesting System model processing

We needed a solution that kept benefits of R, but accelerated it transparently

Zircon Software Overview

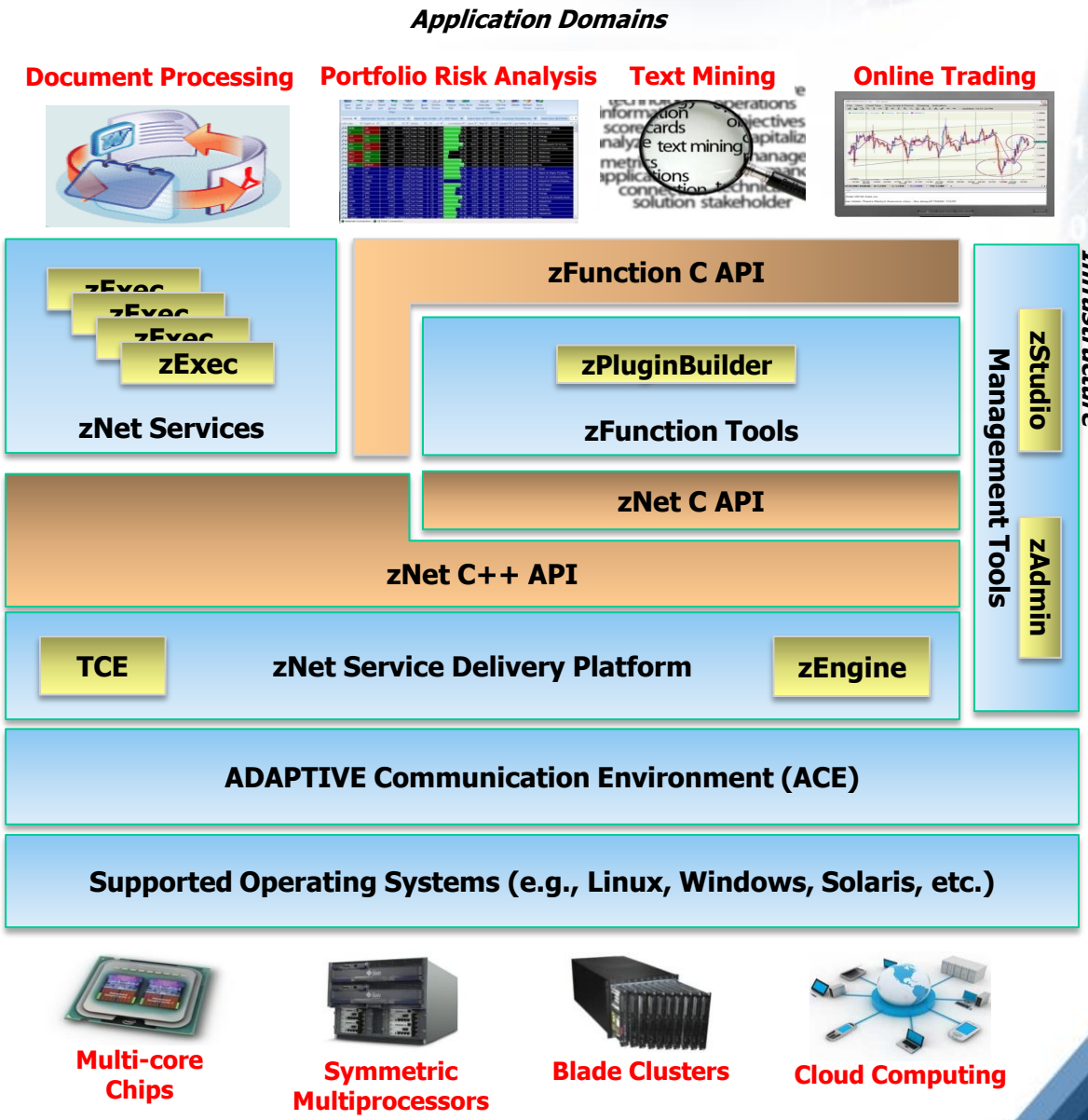
The Zircon software product suite is adaptive middleware

that maps mission-critical applications

onto high-performance computing platforms

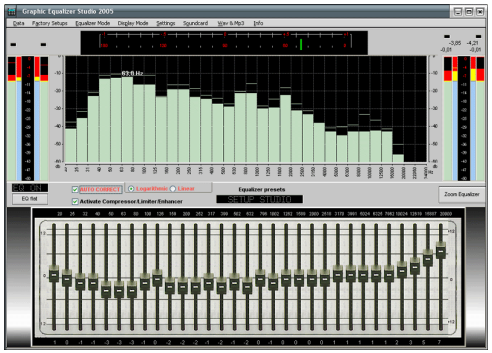
and supports key computing and communication models:

- ***Application executable parallelism***
- ***Function parallelism***
- ***Task parallelism***



Zircon Software Overview

At the heart of the Zircon middleware is a dynamic equalizer that adaptively balances load to ensure scalable and real-time response



Application Domains

Document Processing | **Portfolio Risk Analysis** | **Text Mining** | **Online Trading**

zExec (zExec, zExec, zExec)

zNet Services

zFunction C API

zPluginBuilder

zFunction Tools

zNet C API

zNet C++ API

TCE | **zNet Service Delivery Platform** | **zEngine**

zStudio

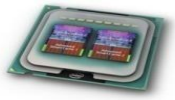
zAdmin

Management Tools

Zircon Middleware and Operating System Infrastructure

ADAPTIVE Communication Environment (ACE)

Supported Operating Systems (e.g., Linux, Windows, Solaris, etc.)



Multi-core Chips

Symmetric Multiprocessors

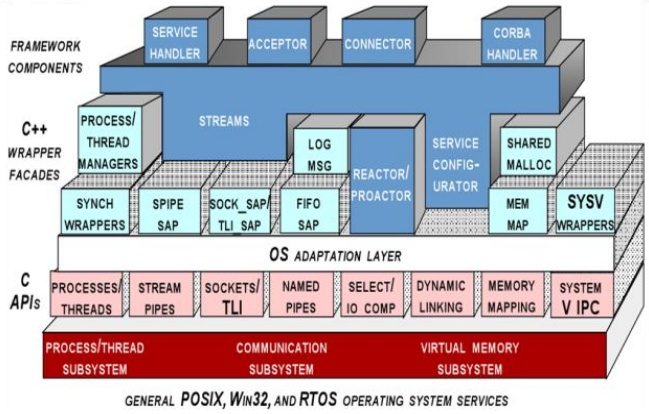
Blade Clusters

Cloud Computing

ZIRCON COMPUTING

Zircon Software Overview

ACE is portable C++ host infrastructure middleware that implements high-performance distributed computing patterns without incurring virtualization overhead



Application Domains

Document Processing | **Portfolio Risk Analysis** | **Text Mining** | **Online Trading**



zExec (zExec, zExec, zExec)

zNet Services

zFunction C API

zPluginBuilder

zFunction Tools

zNet C API

zNet C++ API

TCE | **zNet Service Delivery Platform** | **zEngine**

zStudio

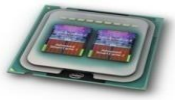
zAdmin

Management Tools

Zircon Middleware and Operating System Infrastructure

ADAPTIVE Communication Environment (ACE)

Supported Operating Systems (e.g., Linux, Windows, Solaris, etc.)


Multi-core Chips


Symmetric Multiprocessors

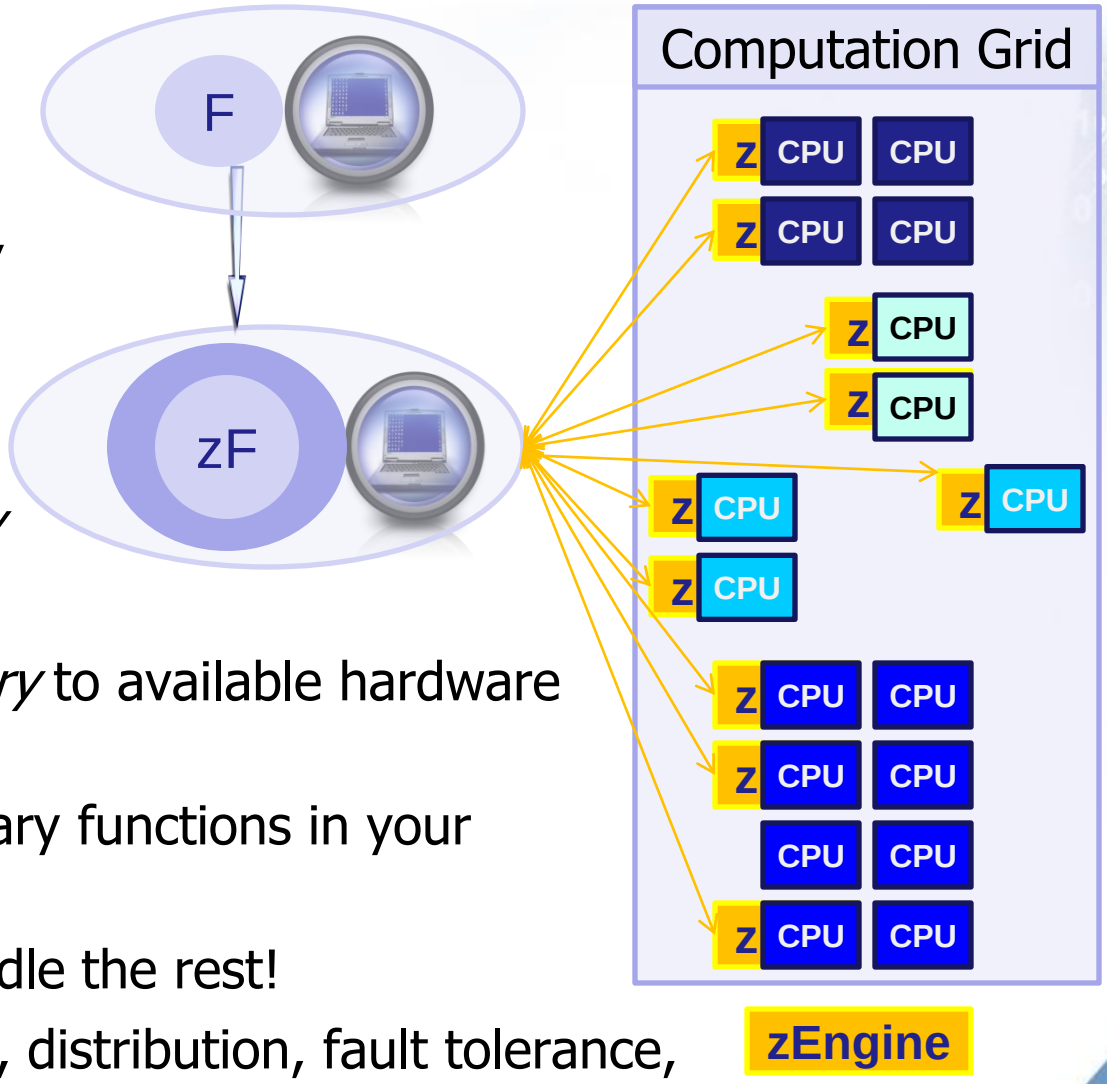

Blade Clusters


Cloud Computing

ZIRCON COMPUTING

How Zircon Software Works

- **Discovery**
  - Designate library that is called repetitively
    - Can be a new, legacy, or 3rd party library
- **Delivery**
  - "zEnable" the library to deliver *zPluginLibrary*
- **Deployment**
  - Deploy the *zPluginLibrary* to available hardware and *zEngine* containers
  - Call zEnabled proxy library functions in your client application
  - Let Zircon software handle the rest!
    - e.g., load balancing, distribution, fault tolerance, concurrency control, security, monitoring, etc.

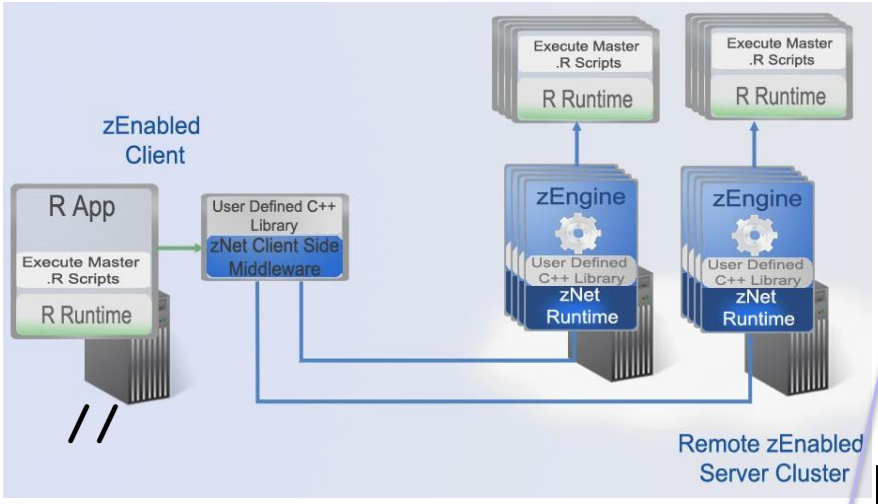


Zircon-based GAM Backtesting System

```
for (x in MODELS) {
   allres <-
      call_get_navs (StratPars)
...
call_get_navs (StratPars, ...) {
  // Use .Call (R capability) to
  // call external C++ function
  fun <- "call_generic_get_navs"
  val <- .Call (fun, StratPars,…)
}
```

```
// This function is written in C++.
// Any R code can call this
// function after loading library
// containing this function.
call_generic_get_navs (StratPars) {
  for (int i = 0;
          i < length (StratPars);
          ++i)
    z_CPP_GenericgetNAVs(...)
}
```

Addressed problem 2 by using Zircon to distribute each row of `StratPars` to zEngine compute servers that execute `GenericgetNAVs()` on input received




```
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
   // Compute-intensive
   // calculations
}
```

Async C++ zAdapterFunction sends each row to `GenericgetNAVs()` R function running on many cores

ZIRCON COMPUTING

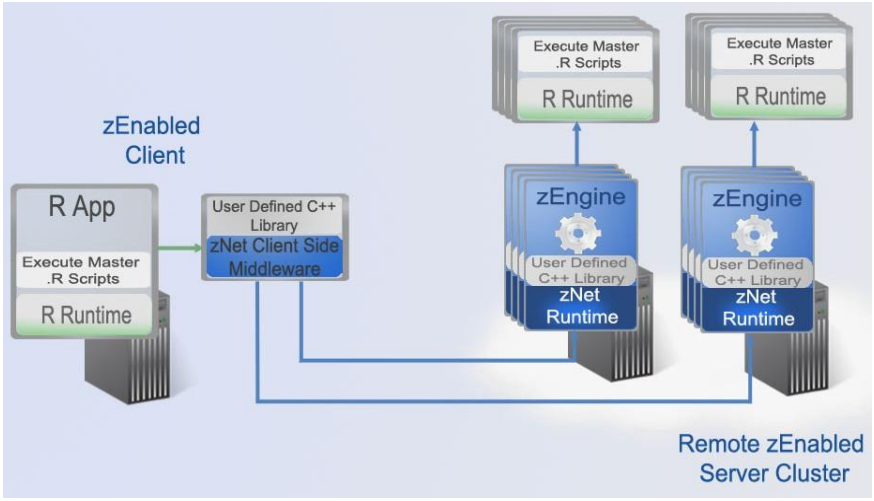
Zircon-based GAM Backtesting System

```
for (x in MODELS) {
   allres <-
      call_get_navs (StratPars)
...
call_get_navs (StratPars, ...) {
  // Use .Call (R capability) to
  // call external C++ function
  fun <- "call_generic_get_navs"
  val <- .Call (fun, StratPars,…)
}
```

```
// This function is written in C++.
// Any R code can call this
// function after loading library
// containing this function.
call_generic_get_navs (StratPars) {
  for (int i = 0;
          i < length (StratPars);
          ++i)
    z_CPP_GenericgetNAVs(...)
}
```

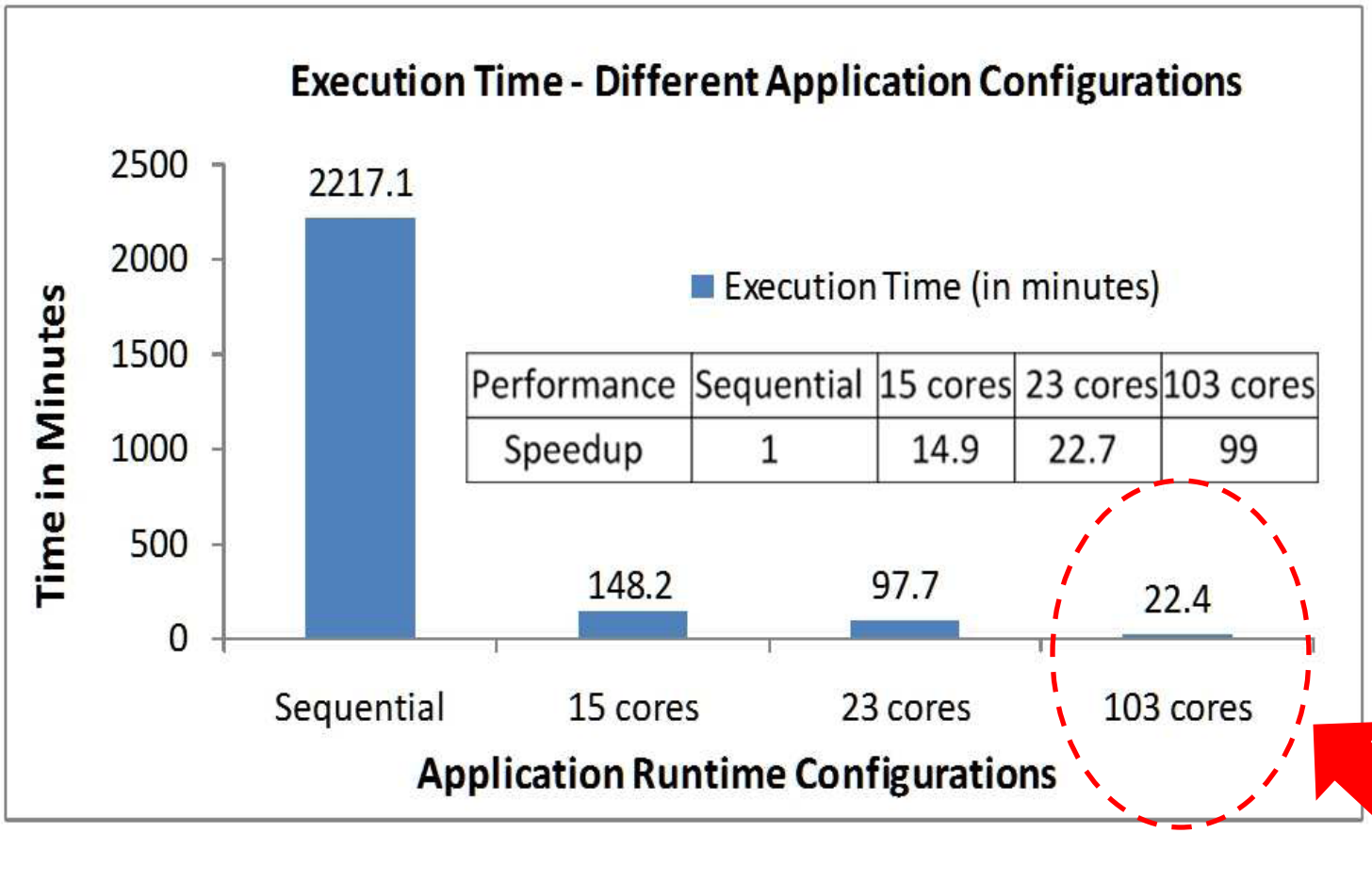
Pros

- Easy to program, evolve, and administer
- Rapid configuration and deployment
- Dynamically scalable and transparently fault-tolerant
- Affordable and cost effective acceleration




```
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
GenericgetNAVs (…) {
   // Compute-intensive
   // calculations
}
```

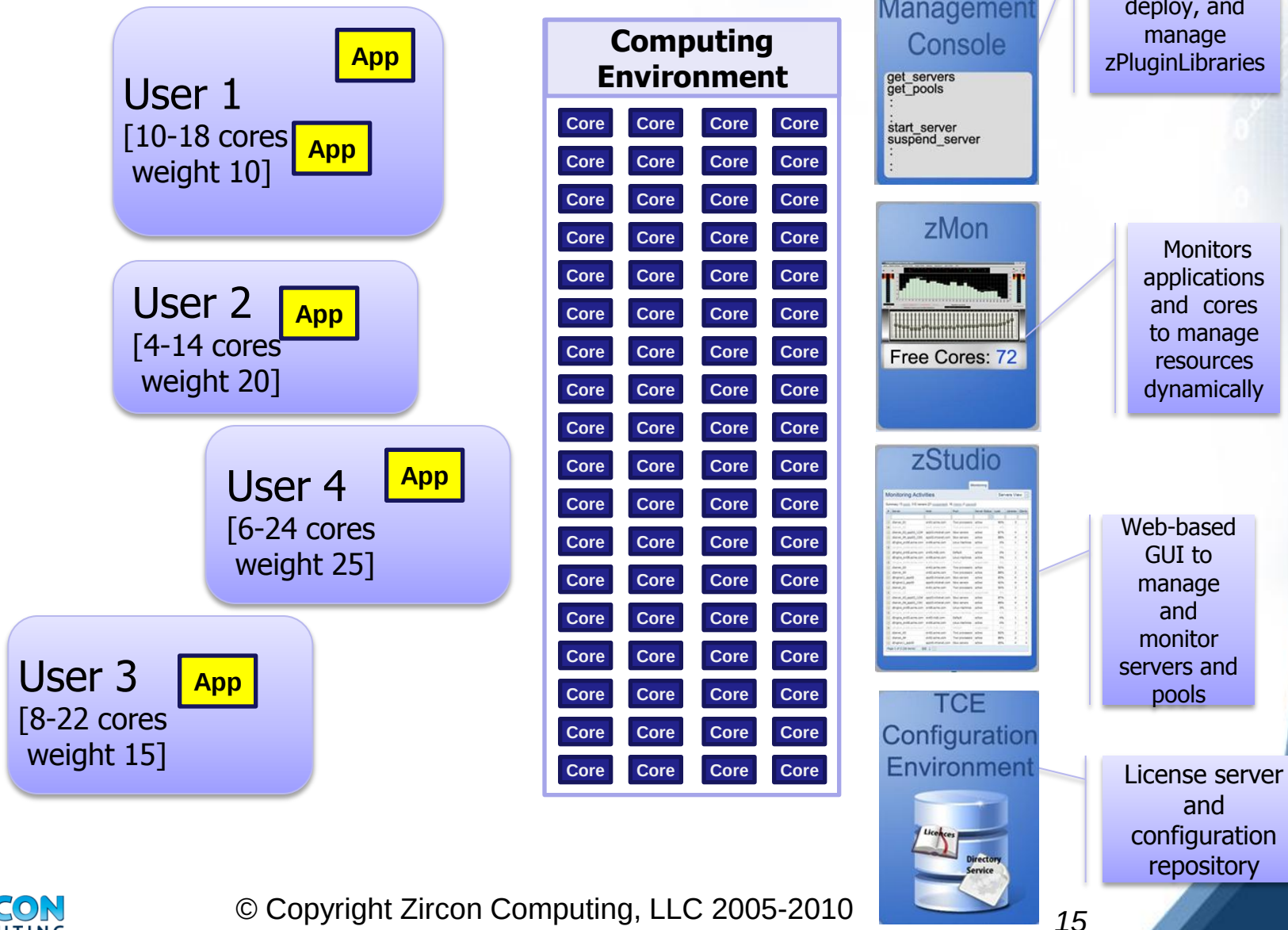
Performance Results Using IBM CoD



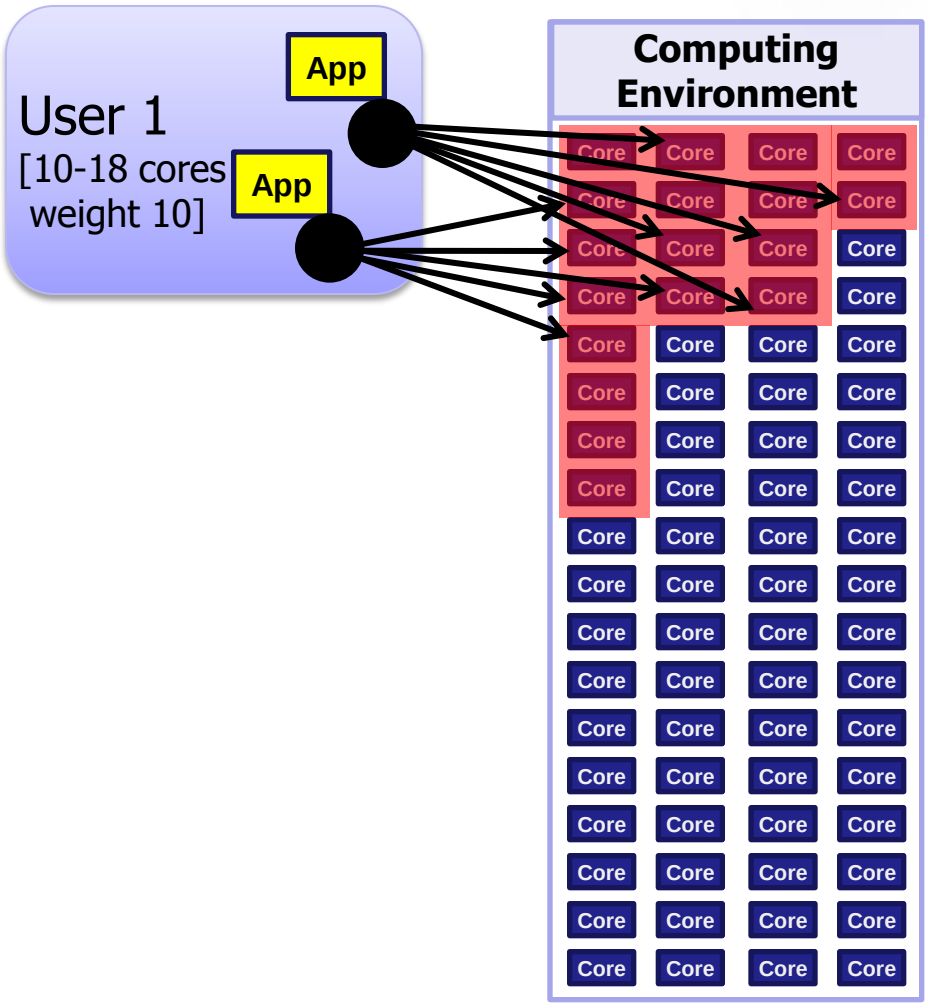
- 13 quad processor/ dual-core (104 cores) 3.0 GHz machines
- Runs 64-bit Red-Hat Enterprise Linux 2.6
- Connected using Gigabit Ethernet

Performance gains by the zEnabled distributed and parallel version of R application limited only by number of cores/machines available to run experiments

Zircon Cloud Enablement Architecture



Zircon Cloud Enablement Architecture



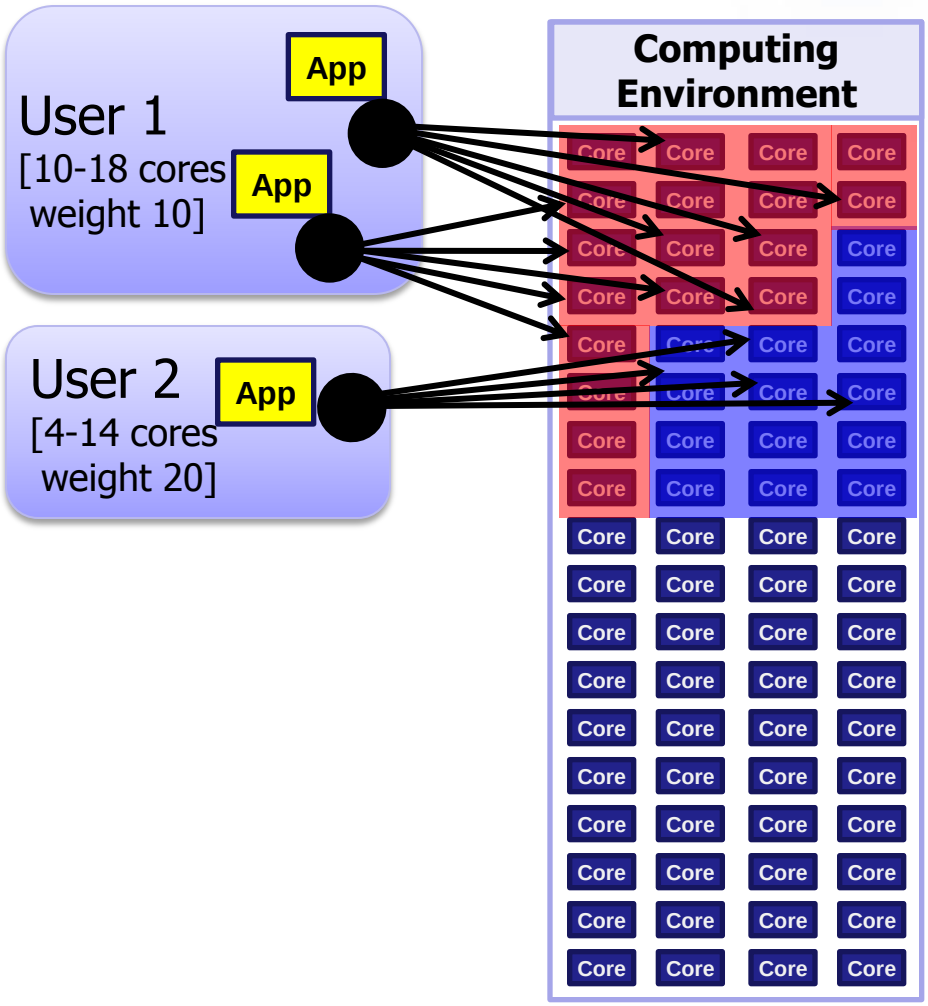
Command-line utility to build, deploy, and manage zPluginLibraries

Monitors applications and cores to manage resources dynamically

Web-based GUI to manage and monitor servers and pools

License server and configuration repository

Zircon Cloud Enablement Architecture



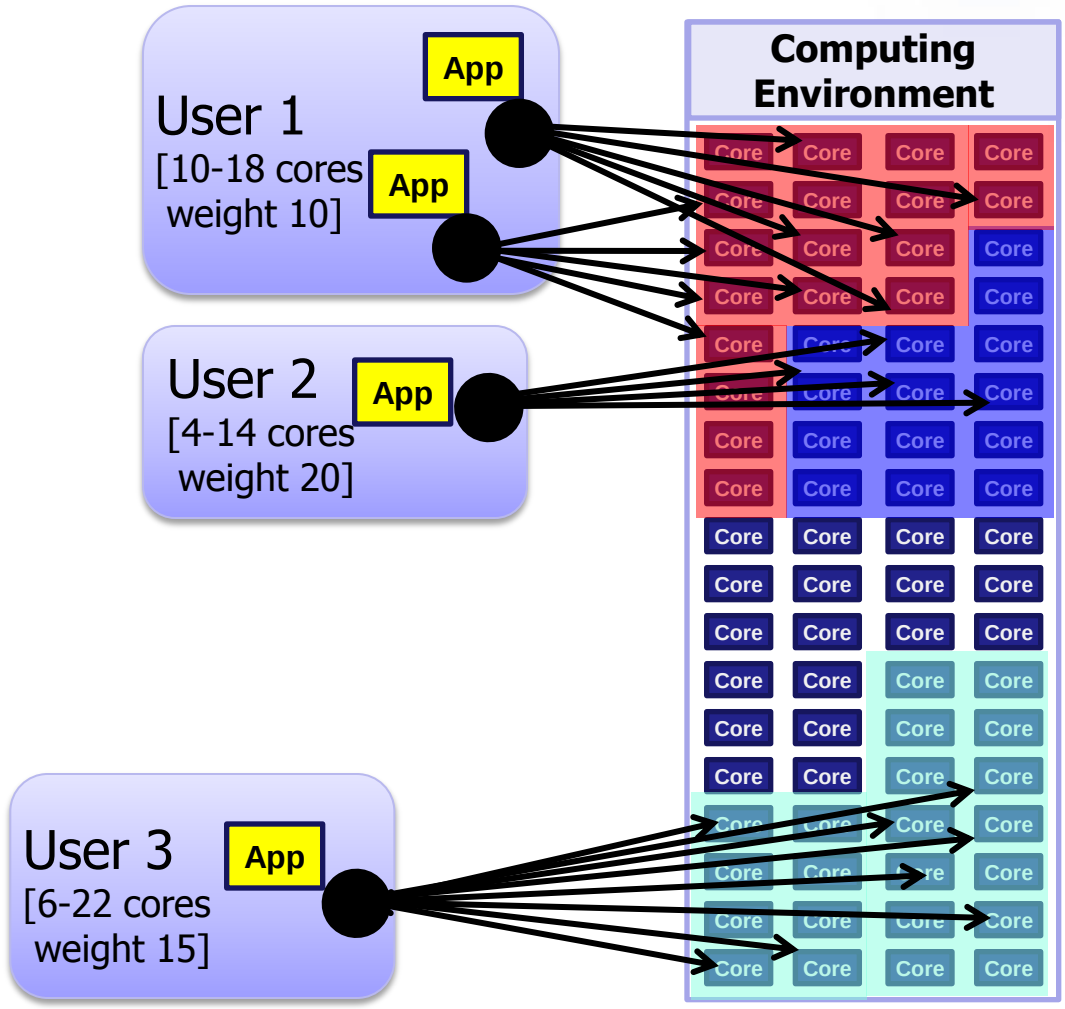
Command-line utility to build, deploy, and manage zPluginLibraries

Monitors applications and cores to manage resources dynamically

Web-based GUI to manage and monitor servers and pools

License server and configuration repository

Zircon Cloud Enablement Architecture



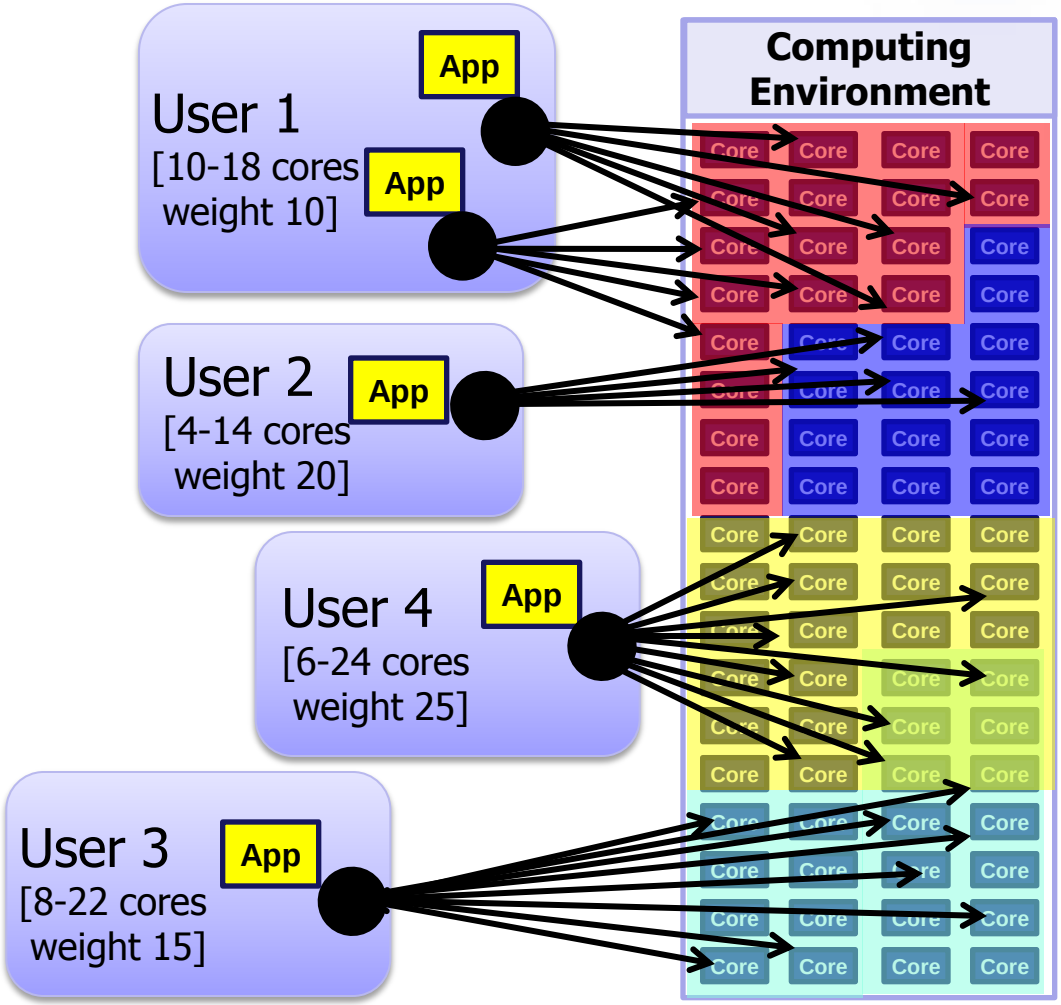
Command-line utility to build, deploy, and manage zPluginLibraries

Monitors applications and cores to manage resources dynamically

Web-based GUI to manage and monitor servers and pools

License server and configuration repository

Zircon Cloud Enablement Architecture



Command-line utility to build, deploy, and manage zPluginLibraries

Monitors applications and cores to manage resources dynamically

Web-based GUI to manage and monitor servers and pools

License server and configuration repository

Zircon Benefits for R-based Apps

Fastest Adaptive Computing Performance

- Real-time Load Equalization
- Transparent Scalability
- Distributed Data Caching
- *UltraFast™* Data Transfer

Minimizes development time for HPC and Cloud Applications

- No Server-Side Development
- Maintains Application Security
- Distributed Data Caching

Quickest to Configure and Deploy

- Automatic Parallel Configuration
- Platform Independence
- Can operate and enable on any cloud, including Amazon
- Can complement/coexist with Hadoop, Map-Reduce, etc. in any cloud

Easiest, Most Intuitive to Use and Deploy

- Automatic Load Equalization
- Automatic Service Discovery
- Automatic Real-time Monitoring and Auditing
- Persistent and Recoverable

Case Study Recap:
Garrett Asset Management: Zircon Enablement

Profile

- International Asset Management Firm specializing in systematic futures, ETFs and currency trading

Objective

- Run 100,000 GAM R-based models two order of magnitude faster than previous processing times

Approach

- Integrated R with Zircon Computing's ultra high performance middleware

Results

- Reduced processing time from 2,217 minutes (36 hours) to 22 minutes (**100x faster)** on commodity ~100 core cloud
- Easy to program, deploy, and accelerate

"Zircon is an expert in the domain of providing increased performance and elastic scalability. Without question they gave us game changing results. The heart of our competitive advantage is our ability to regularly backtest new and existing models. By delivering to us the capability to quickly evaluate our R-based models, it provides me and my investors with valuable information to remain competitive today."

Dr. Elliot Noma CEO and Founder, Garrett Asset Management