
Eagle: Maturation and Evolution
17th Annual Tcl Conference
Joe Mistachkin
EAGLE
BETA V 1.0

Two years ago, at a conference far far away…

- Eagle was presented and there was much rejoicing.
- However, it was not all double-rainbows and flying unicorns.

What is Eagle?

- Eagle is an open-source implementation of Tcl written in C# for the CLR.
- Designed to be highly extensible and easily embeddable into CLR-based applications.
- Supports approximately 90% of the Tcl 8.4 core command set.

What is Eagle, really?

- Originally designed for the purpose of providing a first-class library for scripting applications written for the CLR.
- All other considerations were secondary, including performance and full Tcl compatibility.

What Eagle is not.

- Not based on the Microsoft Dynamic Language Runtime (DLR).
- Not intended for stand-alone application development.
- Unlikely to ever have a compiler.
- Not a replacement for Tcl or Jacl.

I hate Microsoft, .NET, C#, etc.

- Ok, but you like Tcl; otherwise, you wouldn't be here.
- Eagle is good for the Tcl community because it exposes Tcl to an audience that would otherwise have little or no exposure to it.

What if 100% Tcl compatibility is required?

- The included wrapper can be used to access native Tcl from managed code.
- Fully discussed in the Eagle 2009 presentation, available for download.

Where is the innovation?

- The “application is always right” attitude.
  - The IHost interface, etc.
- One interpreter, multiple threads, safely.
- The “universal option parser”.
- The seamless integration with CLR objects.
  - Full support for overload resolution, properties, methods, fields, and events.
- The built-in debugging support.
- Simple development / deployment model (i.e. the “add a reference and go experience”).

Compatibility, performance, and me.

- There are still several major Tcl 8.4 features missing.
- The overall performance is still several orders of magnitude slower than Tcl.
- I’m still the only person working on the project.

What is still missing?

- No Tk.
- No argument expansion syntax (i.e. {*}).
- No namespace support (except the global namespace).
- No asynchronous input/output.
- No [binary], [fblocked], [fileevent], [format], [glob], [history], [memory], [scan], or [trace] commands.
- For [open], command pipelines and serial ports are not supported.
- For [exec], Unix-style input/output redirection and command pipelines are not supported.
- No Safe Base.
- No [registry] or [dde] commands.
- No http, msgcat, or tcltest packages.

Why no namespaces?

- Not enough time in the original schedule.
- It makes command and variable resolution far more complex.

But, I really need namespaces.

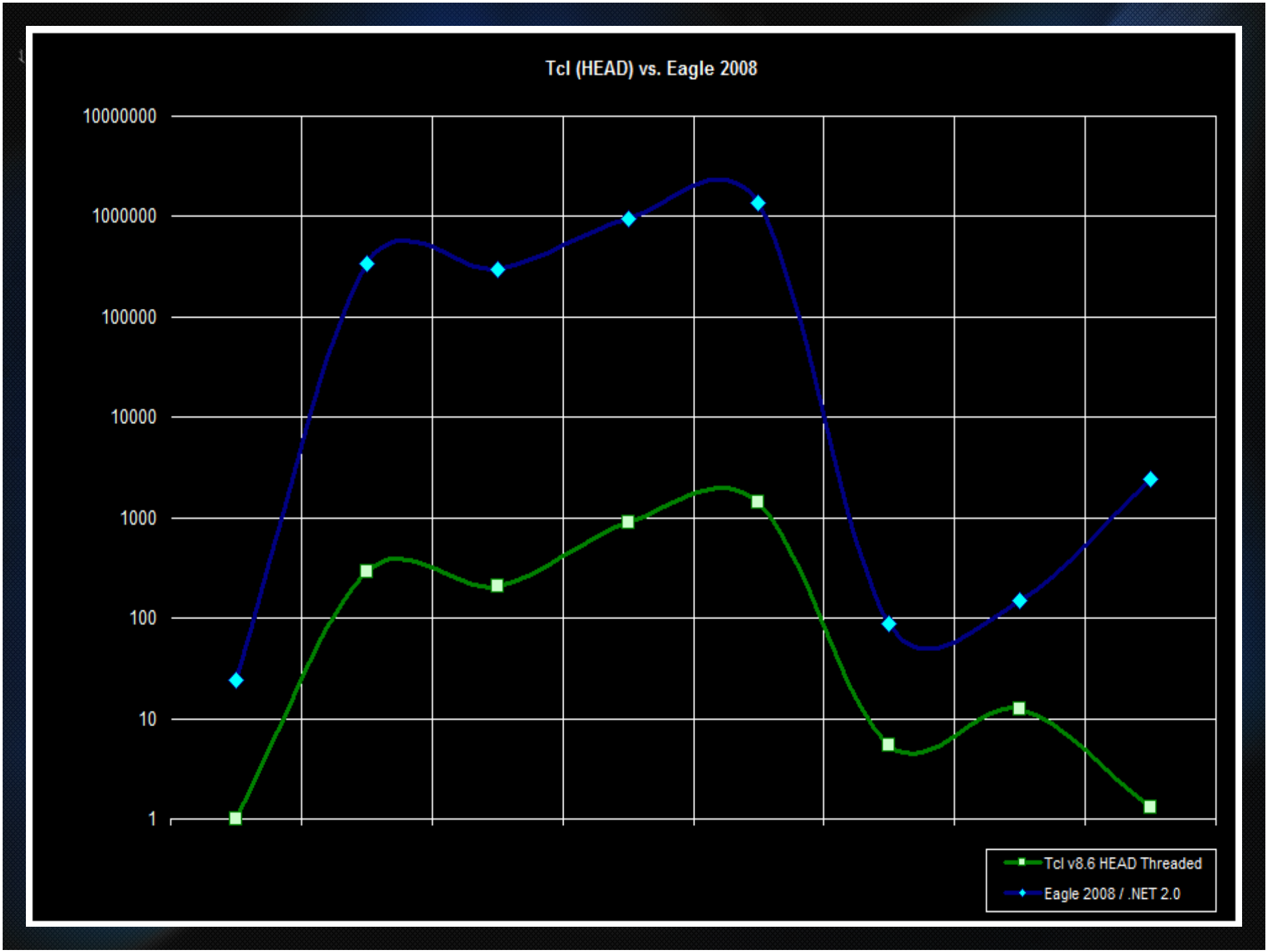
- What steps have already been taken?
- Support for custom resolvers has already been added via the “default resolver” and the managed resolver API.

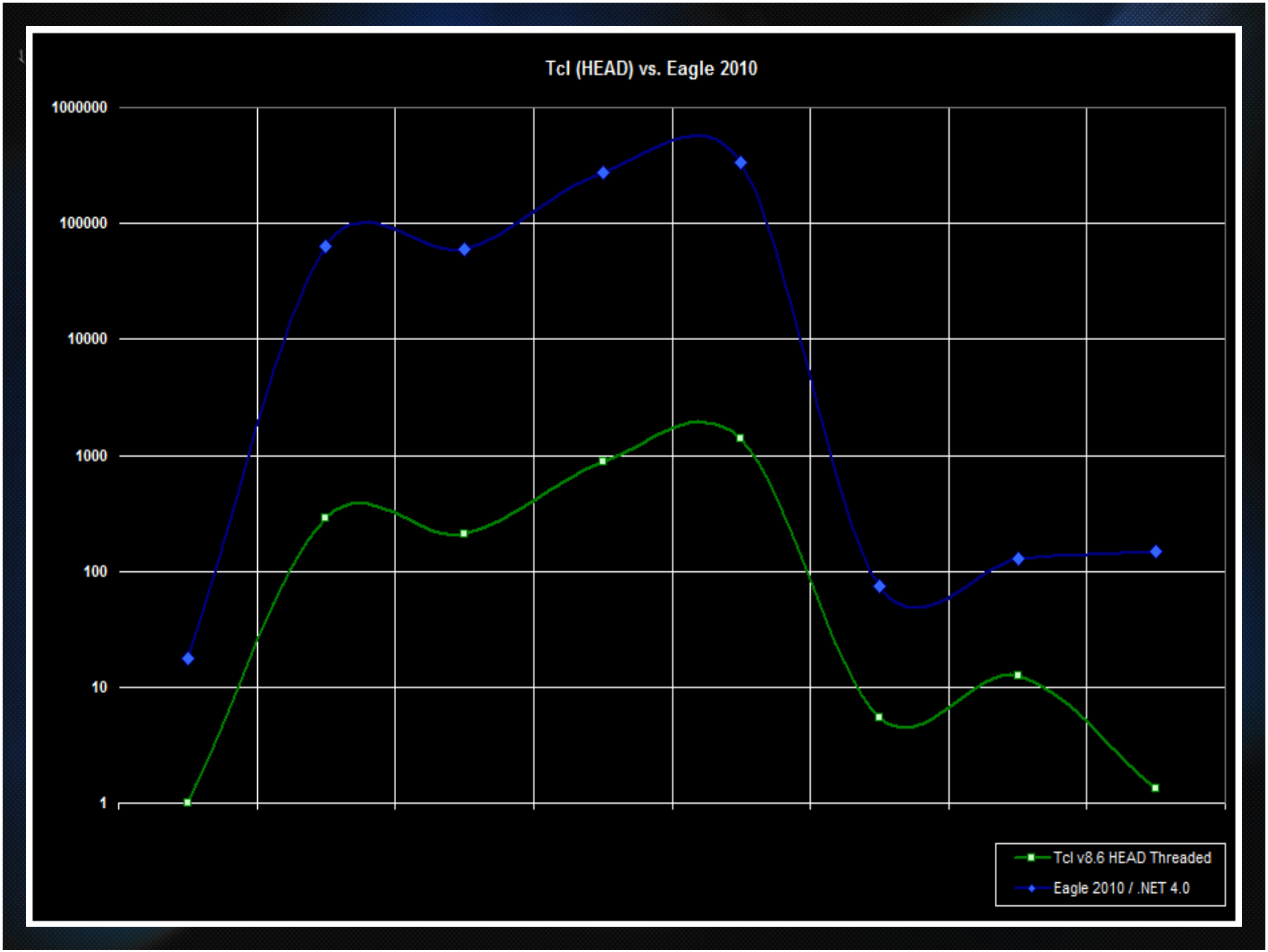
Why no compiler?

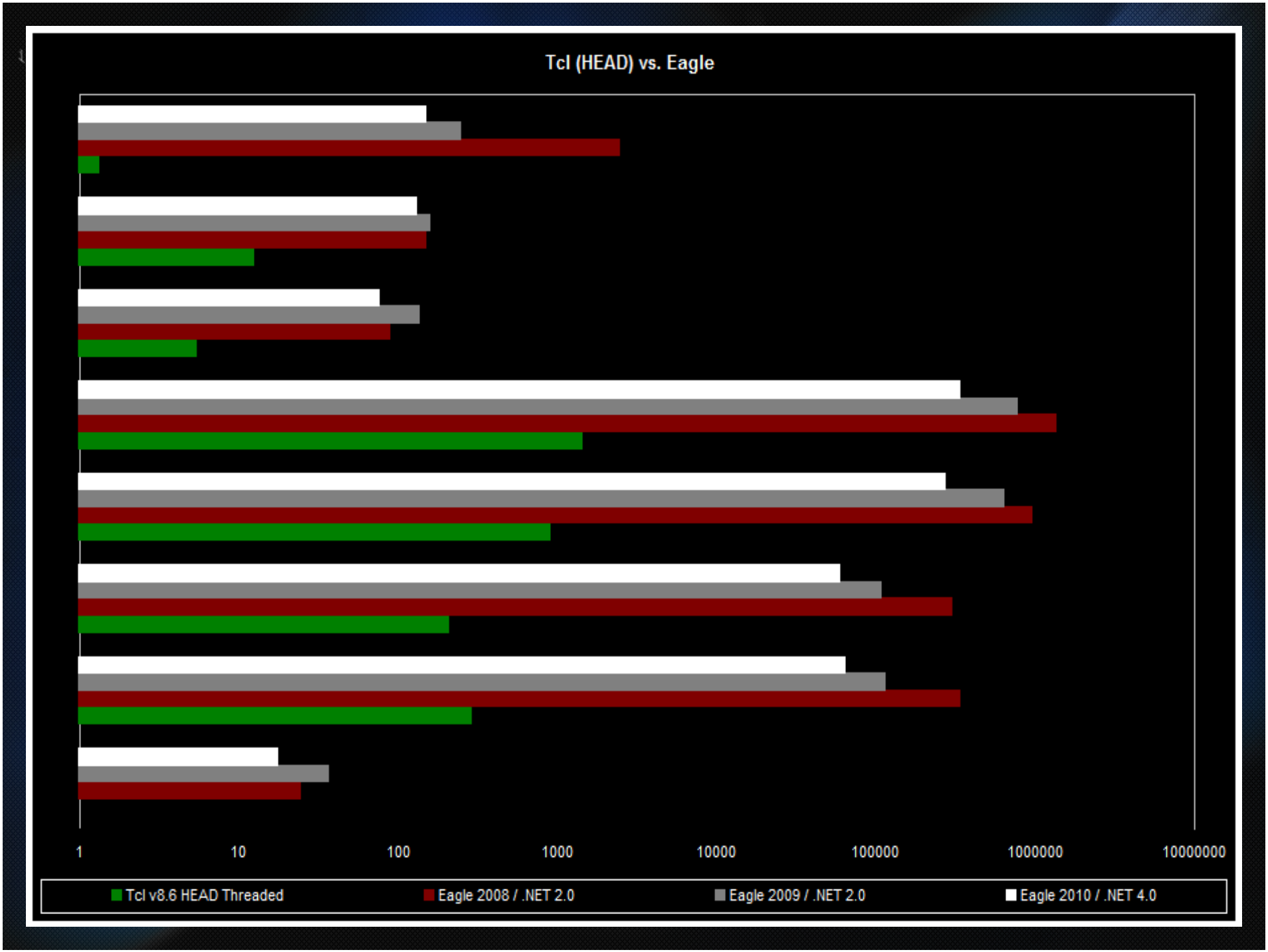
- Not enough time in the original schedule.
- Raw performance was not a primary consideration.
- Being dynamic and correct is more important than being fast.
- Long running scripts can be evaluated (and canceled as necessary) in secondary threads.
- The CLR just-in-time compiler is already pretty good.

Performance Problems

- Can be much slower than native Tcl, even for the simplest operations.
- Over time, targeted optimizations have been added for all critical code paths.

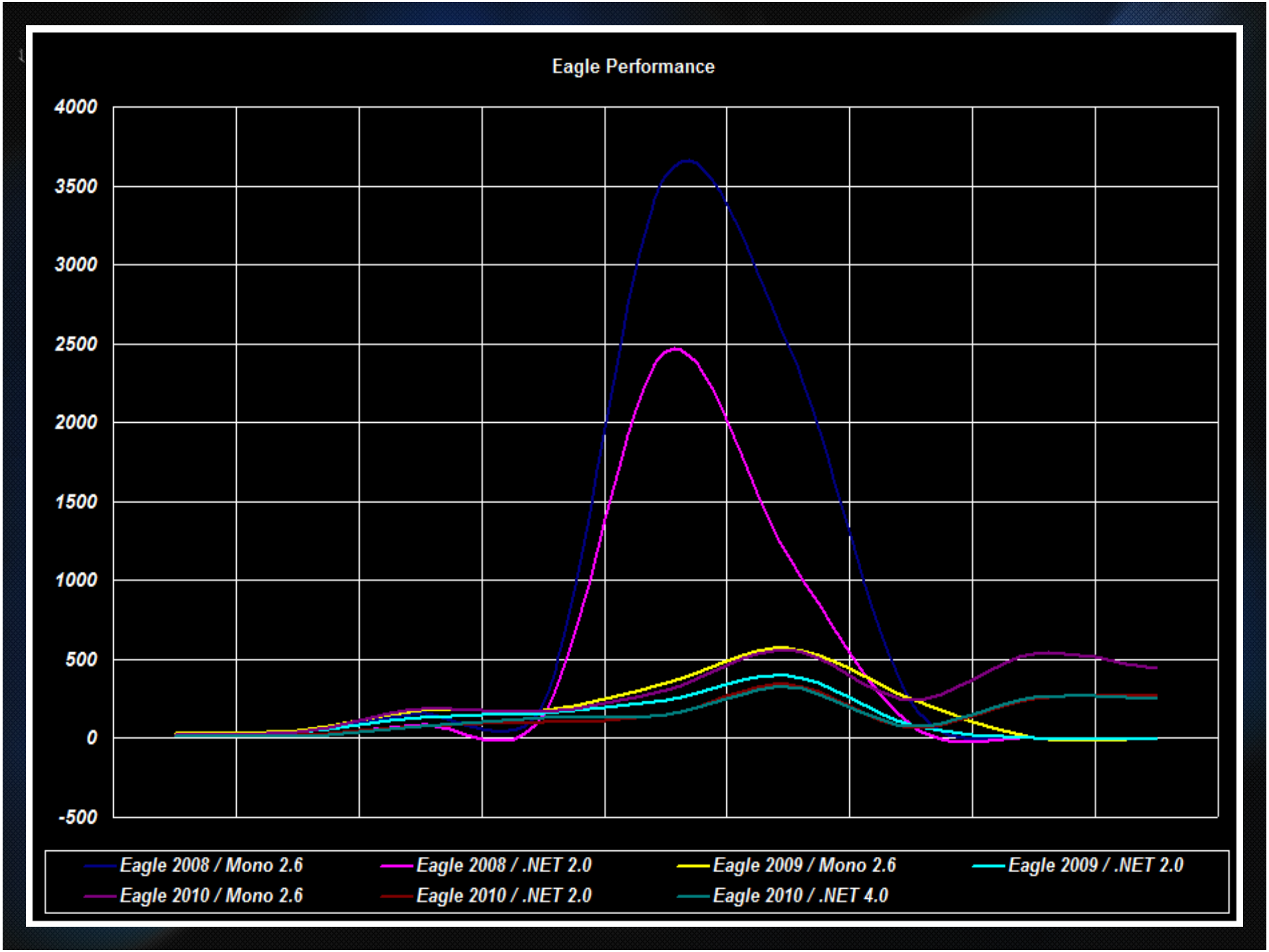

Tcl (HEAD) vs. Eagle 2008
10000000
1000000
100000
10000
1000
100
10
1
Tcl v8.6 HEAD Threaded
Eagle 2008 / .NET 2.0

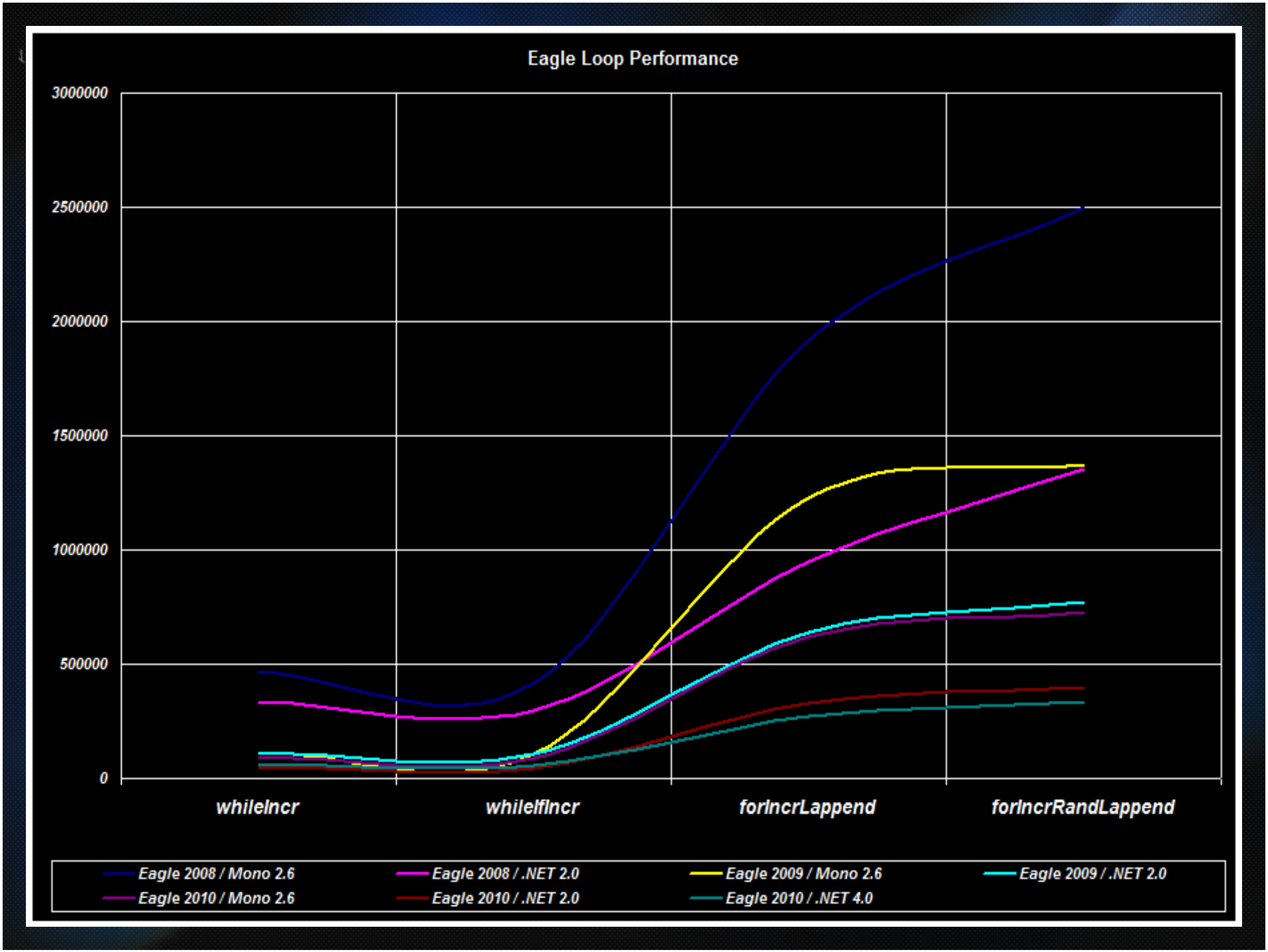

Tcl (HEAD) vs. Eagle 2010
1000000
100000
10000
1000
100
10
1
Tcl v8.6 HEAD Threaded
Eagle 2010 / .NET 4.0

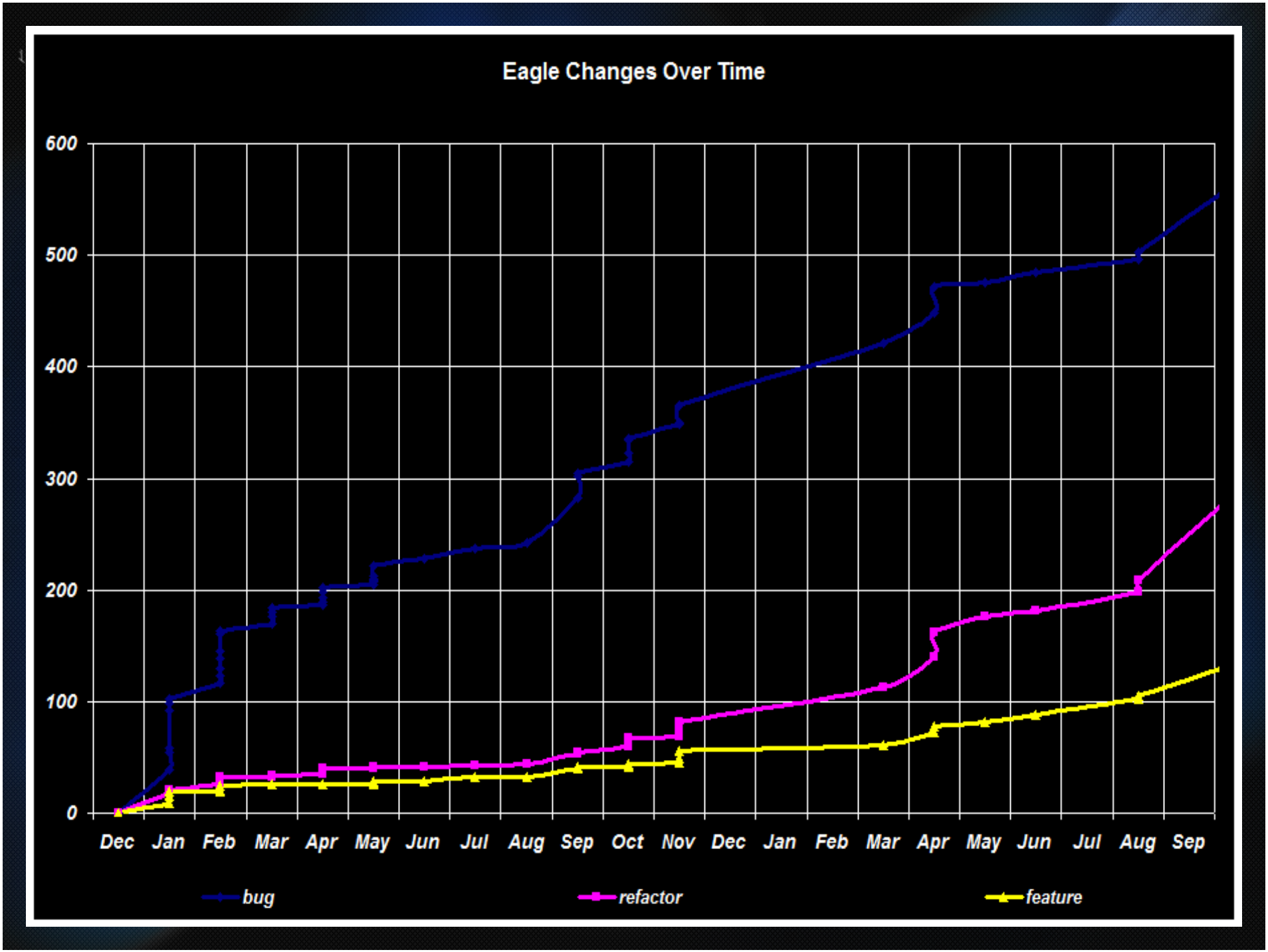

Tcl (HEAD) vs. Eagle
1
10
100
1000
10000
100000
1000000
10000000
Tcl v8.6 HEAD Threaded
Eagle 2008 / .NET 2.0
Eagle 2009 / .NET 2.0
Eagle 2010 / .NET 4.0

What is slow?

- Parsing strings into lists.
- Building lists from strings.
- Expression evaluation, primarily string-to-type conversions.
- All other performance issues are insignificant compared to these three.


Eagle Performance
4000
3500
3000
2500
2000
1500
1000
500
0
-500
Eagle 2008 / Mono 2.6
Eagle 2008 / .NET 2.0
Eagle 2009 / Mono 2.6
Eagle 2009 / .NET 2.0
Eagle 2010 / Mono 2.6
Eagle 2010 / .NET 2.0
Eagle 2010 / .NET 4.0


Eagle Loop Performance
3000000
2500000
2000000
1500000
1000000
500000
0
whileIncr
whileIfIncr
forIncrLappend
forIncrRandLappend
Eagle 2008 / Mono 2.6
Eagle 2008 / .NET 2.0
Eagle 2009 / Mono 2.6
Eagle 2009 / .NET 2.0
Eagle 2010 / Mono 2.6
Eagle 2010 / .NET 2.0
Eagle 2010 / .NET 4.0


Eagle Changes Over Time
600
500
400
300
200
100
0
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
bug
refactor
feature

Architectural Problems, Part 1

- The interpreter class is far too large.
  - Break into multiple files.
  - Move all entity management to a new component.
  - Move all shared state and its management to a new component.
  - Move all “helper” functionality to other classes unless they need access to interpreter internals.
  - The interactive loop code is too large.

Major Components

Engine

- Script / File Evaluation
- Text / File Substitution
- Token Evaluation
- Command Execution
- Script Error Handling
- Host Integration
- Asynchronous Support
- Debugger Support

Parser

- Script Parser
- List Parser / Builder
- String Match “Parser”

Option Parser

Object Marshaller

Interpreter

“Entities”

- Resolvers
- Procedures
- Commands
- Operators
- Functions
- Other

- Test Infrastructure
- Static “Helpers”

Thread State

- Engine
- Interactive
- Variable
- Test

- Interactive Loop
- Global Call Frame
- Other Local / Shared State

- Expression Parser
- Expression Evaluator

Value

- String Conversions
- Operand Handling
- Number / Variant

Host Integration

Tcl/Tk Integration

Architectural Problems, Part 2

- The engine is too recursive.
  - There would potentially be a lot of benefits from something like NRE.
- The components are too tightly coupled and have some circular dependencies.

The Mono Saga

- Mono support was added prior to the conference last year.
- It has never been perfect because of serious bugs in the Mono platform.
- Eagle should build and run correctly on recent versions of Mono (e.g. 2.6.7 and 2.8) for Windows and Unix.

How can I help?

- Test in your environments and report any issues you find.
- Contribute to the documentation and/or the test suite.
- Provide feedback, suggest features, or flames.


Where is it?
https://eagle.to/


Questions and Answers