



IBM Research

# TuningFork: Visualization, Analysis, and Debugging of Complex Real-time Systems

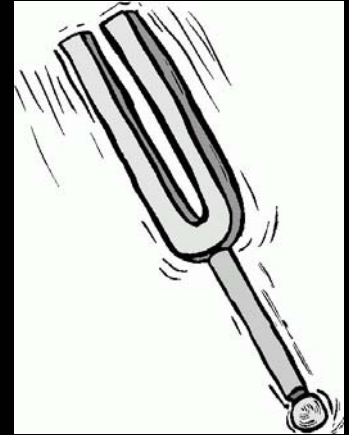
**Bacon, Cheng, Frampton and Grove**

# Metronome Project Overview

- **Make standard Java suitable for real-time programming**
  
- **Real-time Garbage Collection**
  - Jikes RVM prototype [POPL'03]
    - Uniprocessor algorithm
    - Worst-case pause 12ms; utilization at least 45%
  - WebSphere Real-Time product (IBM J9 virtual machine)
    - Worst-case pause 1.5ms; utilization at least 70%
    - Multiprocessor algorithm
    - Version 1 shipped 8/2006
  
- **Application test beds**
  - Harmonicon: Java synthesizer
  - JAviator: Java-controlled helicopter
  - AirJava: Autonomous airplanes

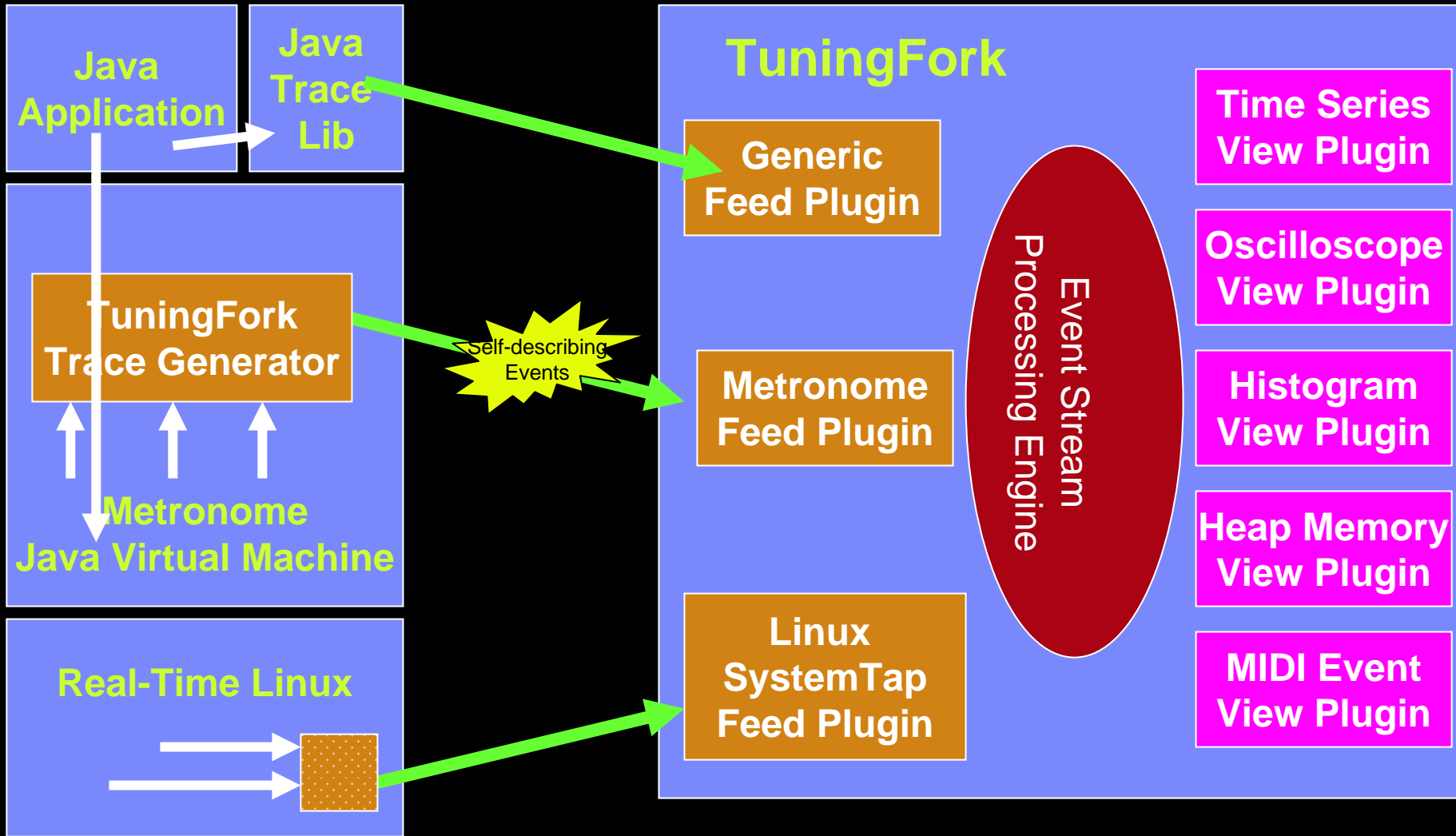
# TuningFork

- **How Do We Debug a 25 MLOC Real Time System????**
- **We care about a SINGLE timing failure**
- **Do we even know there was a failure?**
- **When did it happen?**
- **Which component took too long?**
  - Hardware?
  - Operating System?
  - Virtual Machine?
  - Library?
  - Application?
  - Network?
- **What caused it?**
  - Interactions between multiple activities?





# TuningFork Architecture





# TuningFork on IBM alphaworks

- **Current version (1.0.3)**
  - Java and C++ trace generation libraries (source)
  - Trivial Java/C++ instrumented applications (source)
  - TuningFork (binary)
    - WebSphere Real-Time
    - Java/C++ trace libraries
- **Future Releases**
  - Linux profiling via SystemTap
  - HPM (currently via OProfile)
- **<http://www.alphaworks.ibm.com/tech/tuningfork>**

# Demo Time...

- **WebSphere Real-Time running javac**
  - JVM trace to introduce available visualizations
- **WebSphere Real-Time running SPECjbb2000**
  - Application & JVM traces to show how TuningFork can be used to understand an application performance problem
- **Trivial App over socket (time permitting)**
  - TuningFork supports online and offline visualization