Cooperative Task Management without Manual Stack Management

or, Event-driven Programming is Not the Opposite of Threaded Programming

Atul Adya, Jon Howell, Marvin Theimer, William J. Bolosky, John R. Douceur

Microsoft Research

Presented by

Li Lei
Agenda

1. **Introduction**
2. Task Management and Stack Management
3. A Hybrid Approach
4. Conclusion
Introduction

• Motivation
  – People think “event-driven” programming is the opposite of “multithreaded” programming.
  – Combination of these two styles has some problems.

• Solution
  – We can combine these two styles, with concerning two issues.
  – Provide an approach to solving these problems.
Agenda

1. Introduction

2. Task Management and Stack Management

3. A Hybrid Approach

4. Conclusion
Task Management

• **Preemptive Task Management**
  – Execution of tasks can interleave on uniprocessor or overlap on multiprocessor
  – Efficient but has race conditions

• **Serial Task Management**
  – Executes tasks without any interleavings
  – No race condition but inefficient, e.g. blocking on I/O

• **Cooperative Task Management**
  – A task code only yields control to other tasks at well-defined point in its execution
  – Preserves the advantages of Preemptive and serial task management
Preemptive

Serial

Cooperative

I/O complete

I/O
Stack Management

• **Automatic Stack Management (ASM)**
  – With ASM, the programmer expresses each complete task as a single procedure in the source language
  – It is associated with the style “multithreaded” programming

• **Manual Stack Management (MSM)**
  – MSM requires a programmer to rip the code for any given task into event handlers
  – Event handlers are procedures that respond to the events
  – It is associated with the style “event-driven” programming
ASM vs. MSM

A procedure \( p \) with ASM

A procedure \( P \) with its two event handlers \( P1, P2 \), by MSM

\[ P() \]

\[ \text{I/O} \]

\[ P1() \]

\[ \text{I/O completes} \]

\[ P2() \]
Based on these issues, we clarify ...
Disadvantages of MSM and ASM

• Disadvantages of MSM
  – Affect on program structures
  – Introduce more rules about semantics
  – Software evolution exacerbates this problem

• Disadvantages of ASM
  – Callee procedure as yielding affects the property of caller procedure
  – Software evolution bring such a problem
Agenda

1. Introduction
2. Task Management and Stack Management
3. A Hybrid Approach
4. Conclusion
A Hybrid Approach...

- **Combination of ASM and MSM**
  - Combines their advantages and avoid their disadvantages
  - In software development, programmers working on a same project can use whatever styles they like

- **Some problems exist in the hybrid approach**
  - The code with MSM calls the code with ASM (MSM calls ASM)
  - The code with ASM calls the code with MSM (ASM calls MSM)
MSM calls ASM

Problem 1

• P is a procedure with MSM, including two event handler P1, and P2.
• F is a procedure with ASM, including I/O operations.
• P1 calls F, and P2 is invoked when F finishes
• They are running at a single thread

Both P and F block on I/O!
Problem 2

- P is a procedure with ASM
- F is a procedure with MSM, and two event handlers, including I/O operations,
- P calls F1, and F2 is invoked when I/O completes

Immediately Return to P once F1 finishes

Second return to P after F2 finishes I/O completion.
How to solve these problems?

Use some glue code that can connect them smoothly...

The approach in the paper uses an adaptor to insert between the code with ASM and the code with MSM such that the codes with different styles can not directly communicate.
MSM calls ASM

Problem 1

- P is a procedure with MSM, including two event handler P1, and P2.
- F is a procedure with ASM, including I/O operations.
- P1 calls F, and P2 is invoked when F finishes.
- They are running at a single thread.
MSM calls ASM

Adaptor fork a new thread and execute F on it.

The control can be schedule back.

F calls back adaptor, with P2, handler.
ASM calls MSM

Problem 2

- P is a procedure with ASM
- F is a procedure with MSM, and two event handlers, including I/O operations,
- P calls F1, and F2 is invoked when I/O completes
ASM calls MSM

Adaptor calls F1 instead of P calls F1.

Adaptor does not return to p, scheduling other threads.

The adaptor calls P finally.
Agenda

1. Introduction
2. Task Management and Stack Management
3. A Hybrid Approach
4. Conclusion
Conclusion

• “Event-driven” is not the opposite of “multithreaded”
  – There is a combination that uses both of them: Cooperative Task Management with automatic stack management

• Some problems are addressed in combination
  – Interactions between ASM and MSM
  – Adaptor can solve this problem by connecting each of them in the middle.
QUESTIONS ?