Reimplementación de la máquina abstracta MAPiCO

Alba Liliana Sarasti Campo Carlos Alberto Llano Rodriguez

Pontificia Universidad Javeriana - Cali

23 de enero de 2007

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,
4. Registros genericos para almacenar cualquier tipo de dato,

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,
4. Registros genericos para almacenar cualquier tipo de dato,
5. Configuración parametrizable por archivo o por instrucción,

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,
4. Registros genericos para almacenar cualquier tipo de dato,
5. Configuración parametrizable por archivo o por instrucción,
6. Uso de plugins para cargar dinamicamente las instrucciones de la máquina,

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,
4. Registros genericos para almacenar cualquier tipo de dato,
5. Configuración parametrizable por archivo o por instrucción,
6. Uso de plugins para cargar dinamicamente las instrucciones de la máquina,
7. Facilidad para adicionar plugins a la maquina por medio de la configuración,

MAPiCO - Implementación

Caracteristicas:

1. Lenguaje de programación C (Estándar ANSI C99),
2. Estructuras de datos modularizadas y genericas (TAD's),
3. Area de memoria, procesos, lista de parametros dinamicos y colas, utilizan TAD's hechos en macros de C que disminuyen los tiempos de ejecución,
4. Registros genericos para almacenar cualquier tipo de dato,
5. Configuración parametrizable por archivo o por instrucción,
6. Uso de plugins para cargar dinamicamente las instrucciones de la máquina,
7. Facilidad para adicionar plugins a la maquina por medio de la configuración,
8. Facilidad para agregar instrucciones a los plugins actualizando el catalogo del plugin,

MAPiCO - Implementación

Caracteristicas :

1. Posibilidad de las instrucciones de agregar o eliminar variables a la lista de parametros dinamicos en tiempo de ejecución,

MAPiCO - Implementación

Caracteristicas :

1. Posibilidad de las instrucciones de agregar o eliminar variables a la lista de parametros dinamicos en tiempo de ejecución,
2. Ejecución de instrucciones diferentes a las del calculo PiCO,

MAPiCO - Implementación

Caracteristicas :

1. Posibilidad de las instrucciones de agregar o eliminar variables a la lista de parametros dinamicos en tiempo de ejecución,
2. Ejecución de instrucciones diferentes a las del calculo PiCO,
3. Cambio del sistema de restricciones modificando solamente la interfaz del store,

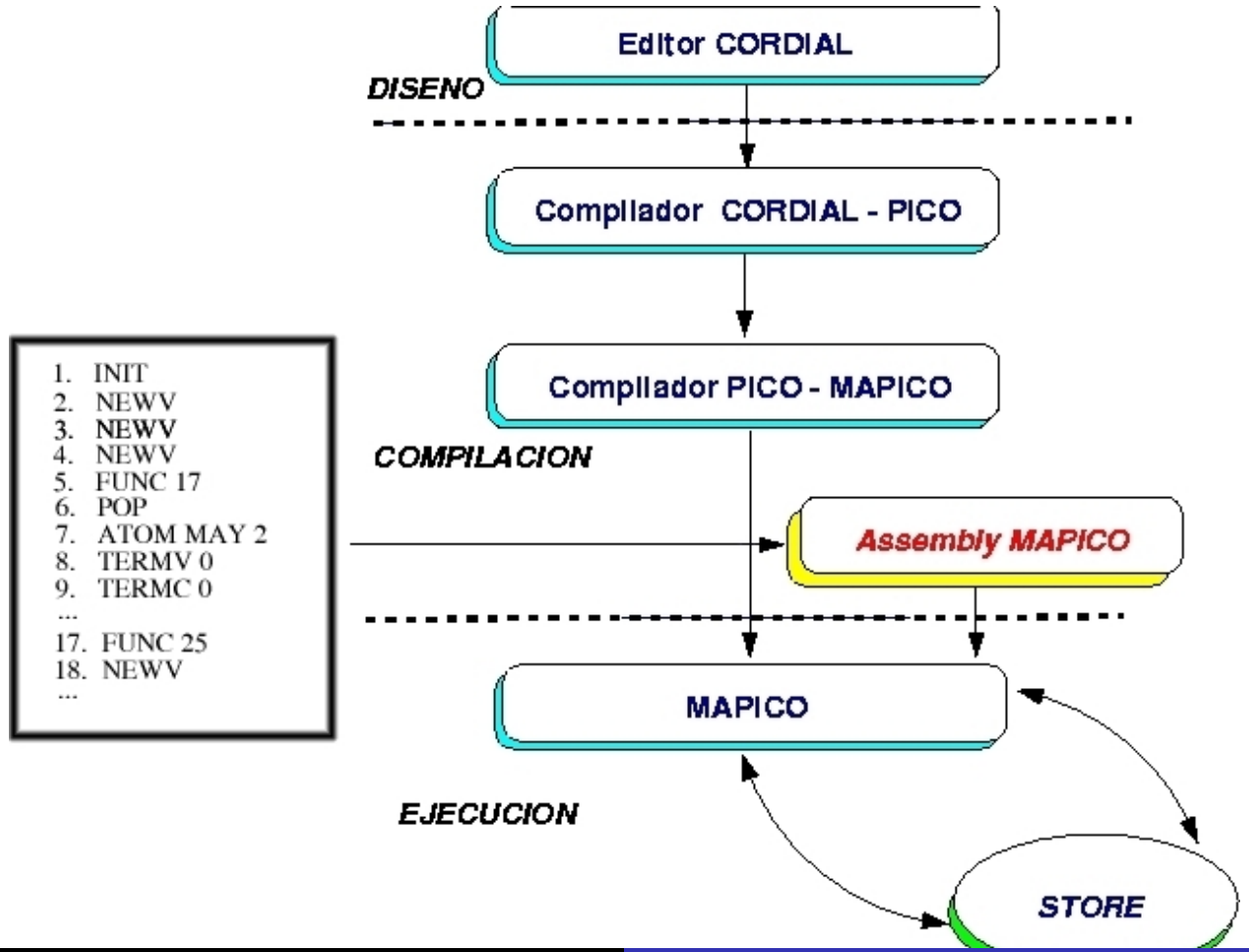
MAPiCO - Implementación

Caracteristicas :

1. Posibilidad de las instrucciones de agregar o eliminar variables a la lista de parametros dinamicos en tiempo de ejecución,
2. Ejecución de instrucciones diferentes a las del calculo PiCO,
3. Cambio del sistema de restricciones modificando solamente la interfaz del store,
4. Desarrollada bajo los estandares establecidos por GNU para el ordenamiento de proyectos lo que asegura la portabilidad de la máquina,

AssemblyMAPiCO

El *Assembly MAPiCO* es un programa que se encarga de traducir, un archivo fuente en formato texto a un archivo de salida en formato binario. Este archivo de salida contiene el código que *MAPiCO* ejecuta (*bytecode*).

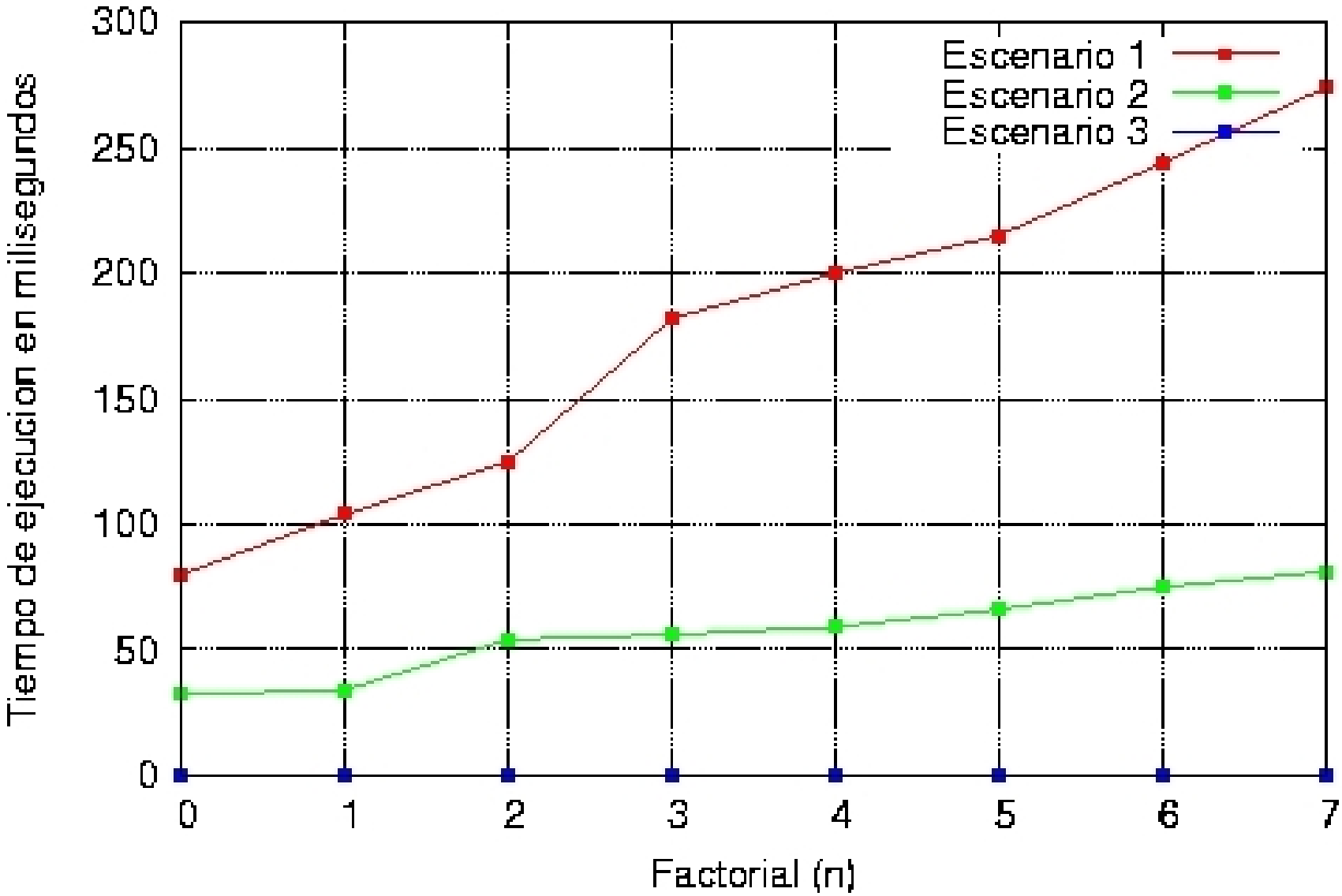


Pruebas y Resultados

Factorial:

Programa Factorial	**Escenarios de Pruebas**		
	Escenario 1 *Máquina JAVA / Store JAVA*	**Escenario 2** *Máquina JAVA / Store C*	**Escenario 3** *Máquina C / Store C*
0	80	32	0.0009
1	104	34	0.003
2	125	54	0.005
3	182	56	0.008
4	200	59	0.011
5	215	66	0.014
6	244	75	0.020
7	274	81	0.023

Factorial


Escenario 1
Escenario 2
Escenario 3
Tiempo de ejecucion en milisegundos
Factorial (n)
0
50
100
150
200
250
300
0
1
2
3
4
5
6
7

SEND + MORE = MONEY

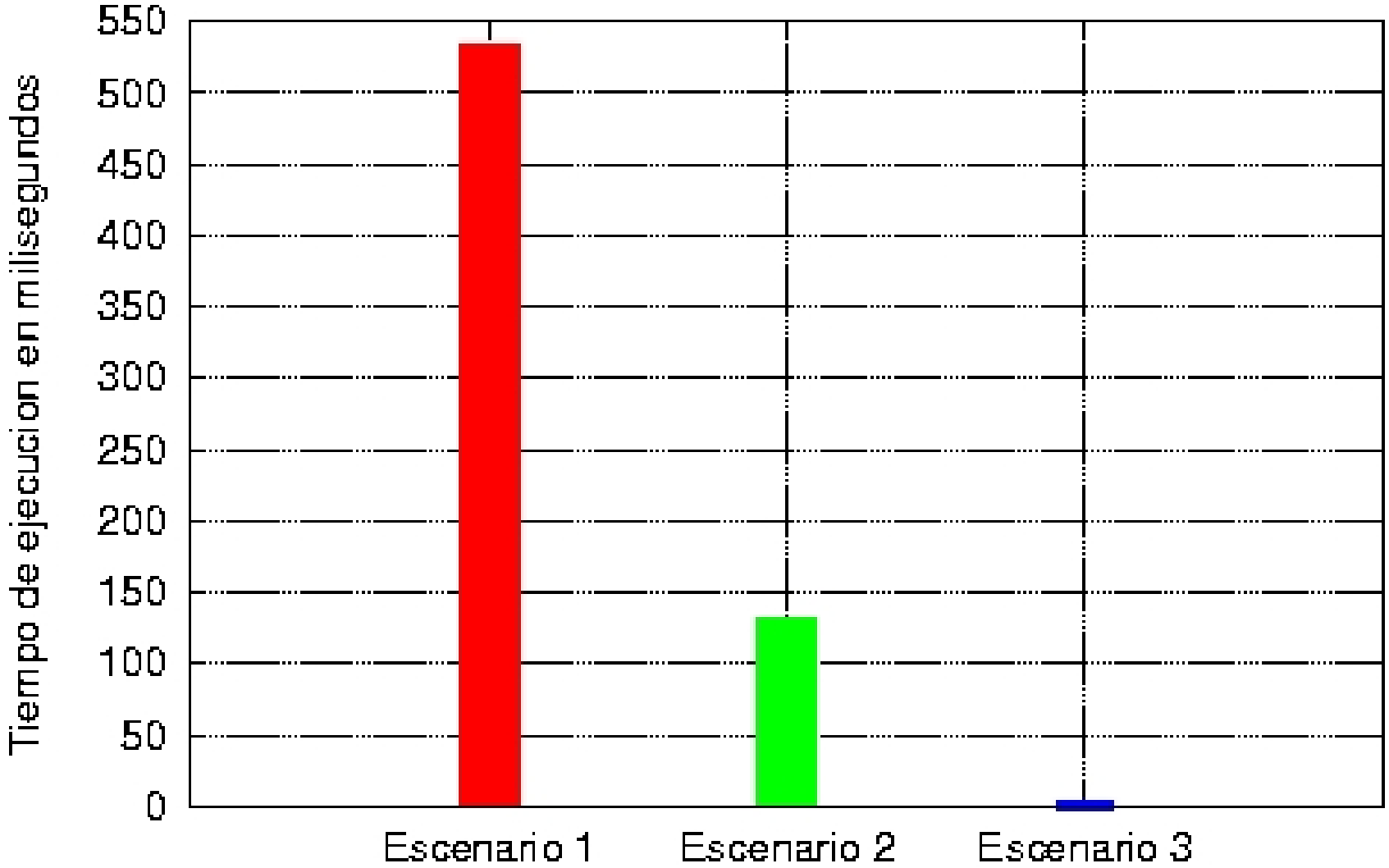
$$\begin{array}{r} \text{SEND} \\ +\text{MORE} \\ \hline \text{MONEY} \end{array} \qquad \begin{array}{r} 9567 \\ +\ 1085 \\ \hline 10652 \end{array}$$

Programa SMM	**Escenarios de Pruebas**		
	Escenario 1 *Máquina JAVA / Store JAVA*	**Escenario 2** *Máquina JAVA / Store C*	**Escenario 3** *Máquina C / Store C*
TELL $Y = 2$	533	132	0.0690

Los resultados para el **Escenario 3** son los siguientes:

$$\begin{array}{lll} S & in & \{9,9\} \\ E,D,R & in & \{3,8\} \\ N & in & \{3,6\} \\ M & in & \{1,1\} \\ O & in & \{0,0\} \\ Y & in & \{2,2\} \end{array}$$

SEND + MORE = MONEY

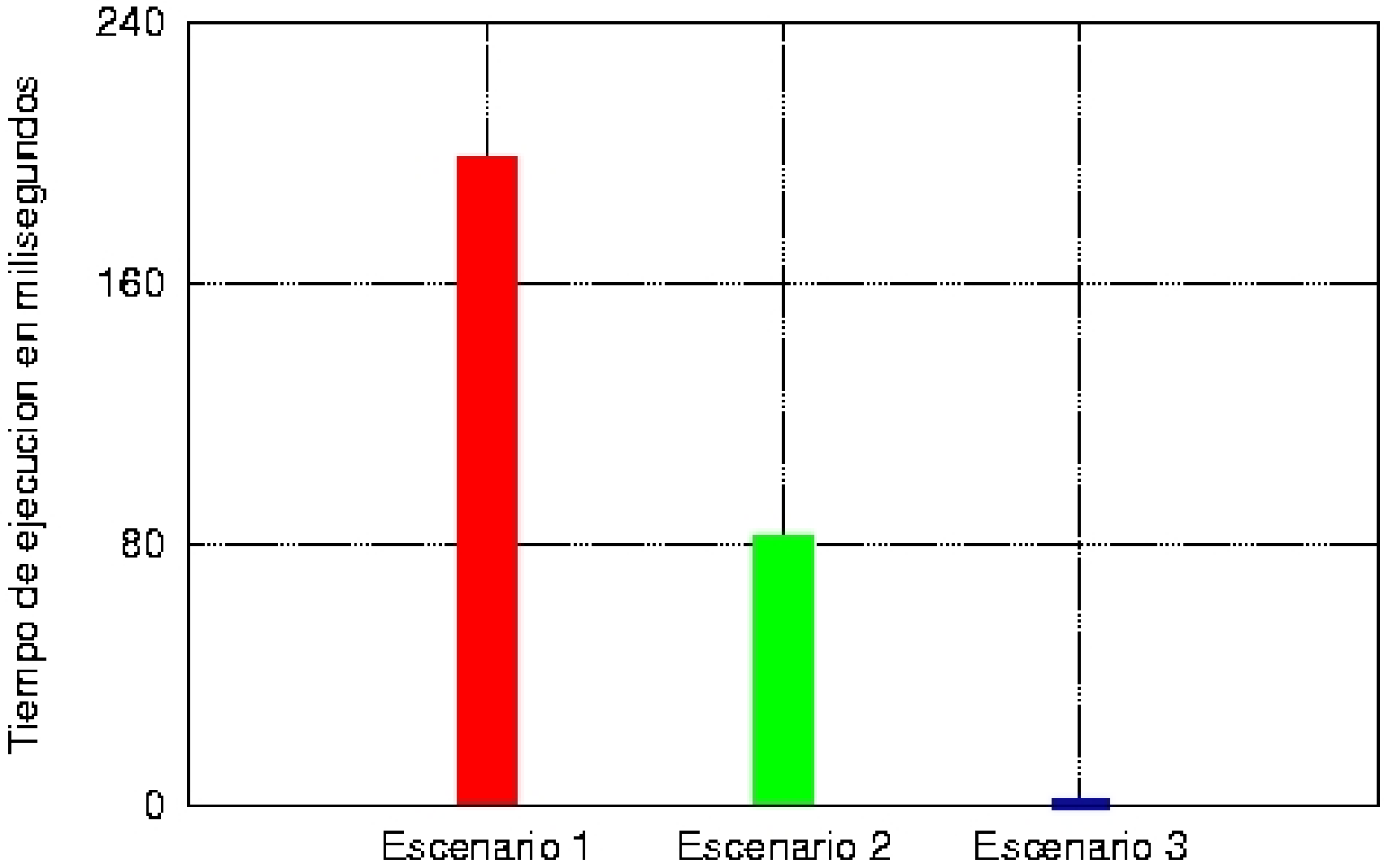


Programa NREINAS	**Escenarios de Pruebas**		
	Escenario 1 *Máquina JAVA / Store JAVA*	**Escenario 2** *Máquina JAVA / Store C*	**Escenario 3** *Máquina C / Store C*
TELL $X_3 = 4$	199	83	0.0229

Los resultados para el **Escenario 3** son los siguientes:

$$\begin{array}{lll} X_1 & in & \{3,3\} \\ X_2 & in & \{1,1\} \\ X_3 & in & \{4,4\} \\ X_4 & in & \{2,2\} \end{array}$$

N-REINAS


240
160
80
0
Tiempo de ejecucion en milisegundos
Escenario 1
Escenario 2
Escenario 3

Objetivos planteados vs resultados obtenidos de acuerdo al anteproyecto

Conclusiones

1. *MAPiCO*, es soportada por el cálculo lógico, correcto y formal *PiCO*

Conclusiones

1. *MAPiCO*, es soportada por el cálculo lógico, correcto y formal *PiCO*
2. Al implementar la maquina en el lenguaje de programación C se garantizaron tiempos de ejecucion mucho mas eficientes,

Conclusiones

1. *MAPiCO*, es soportada por el cálculo lógico, correcto y formal *PiCO*
2. Al implementar la maquina en el lenguaje de programación C se garantizaron tiempos de ejecucion mucho mas eficientes,
3. Su diseño modular permite que ante cambios en el calculo PiCO las modificaciones solo se hagan del lado de los plugins y no deba modificarse la máquina abstracta,

Conclusiones

1. *MAPiCO*, es soportada por el cálculo lógico, correcto y formal *PiCO*
2. Al implementar la maquina en el lenguaje de programación C se garantizaron tiempos de ejecucion mucho mas eficientes,
3. Su diseño modular permite que ante cambios en el calculo PiCO las modificaciones solo se hagan del lado de los plugins y no deba modificarse la máquina abstracta,
4. Debido a la forma de implementación, se puede implementar la funcionalidad de otra máquina en plugins y ejecutarlos en MAPiCO, la Máquina Abstracta *LMAN* podría implementarse bajo este criterio,

Conclusiones

1. *MAPiCO*, es soportada por el cálculo lógico, correcto y formal *PiCO*
2. Al implementar la maquina en el lenguaje de programación C se garantizaron tiempos de ejecucion mucho mas eficientes,
3. Su diseño modular permite que ante cambios en el calculo PiCO las modificaciones solo se hagan del lado de los plugins y no deba modificarse la máquina abstracta,
4. Debido a la forma de implementación, se puede implementar la funcionalidad de otra máquina en plugins y ejecutarlos en MAPiCO, la Máquina Abstracta *LMAN* podría implementarse bajo este criterio,
5. La alternativa de eliminación de variables planteada podría significar una mejora considerable en el rendimiento del *Store*,

Conclusiones

1. Se realizaron todo tipo de pruebas sobre las estructuras de datos que utiliza *MAPiCO* garantizando el optimo rendimiento de la máquina.

Conclusiones

1. Se realizaron todo tipo de pruebas sobre las estructuras de datos que utiliza *MAPiCO* garantizando el optimo rendimiento de la máquina.
2. Las pruebas realizadas demostraron que:

Conclusiones

1. Se realizaron todo tipo de pruebas sobre las estructuras de datos que utiliza *MAPiCO* garantizando el optimo rendimiento de la máquina.
2. Las pruebas realizadas demostraron que:
   - El **Escenario** determinado por *MAPICO JAVA* y *Store C* es un 66 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store JAVA*.

Conclusiones

1. Se realizaron todo tipo de pruebas sobre las estructuras de datos que utiliza *MAPiCO* garantizando el optimo rendimiento de la máquina.
2. Las pruebas realizadas demostraron que:
   - El **Escenario** determinado por *MAPICO JAVA* y *Store C* es un 66 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store JAVA*.
   - El **Escenario** determinado por *MAPICO C* y *Store C* es un 99 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store JAVA*.

Conclusiones

1. Se realizaron todo tipo de pruebas sobre las estructuras de datos que utiliza *MAPiCO* garantizando el optimo rendimiento de la máquina.
2. Las pruebas realizadas demostraron que:
   - El **Escenario** determinado por *MAPICO JAVA* y *Store C* es un 66 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store JAVA*.
   - El **Escenario** determinado por *MAPICO C* y *Store C* es un 99 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store JAVA*.
   - El **Escenario** determinado por *MAPICO C* y *Store C* es un 99 % más eficiente que el **Escenario** constituido por *MAPICO JAVA* y *Store C*.

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,
5. Implementar un Explorador de Restriciones similar al trabajo de grado *Explorador de Cordial*

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,
5. Implementar un Explorador de Restriciones similar al trabajo de grado *Explorador de Cordial*
6. Implementar el Compilador *PiCO - MAPiCO* tratando de optimizar la generación de variables,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,
5. Implementar un Explorador de Restriciones similar al trabajo de grado *Explorador de Cordial*
6. Implementar el Compilador *PiCO - MAPiCO* tratando de optimizar la generación de variables,
7. Implementar un Garbage Collector dentro de *MAPiCO*,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,
5. Implementar un Explorador de Restriciones similar al trabajo de grado *Explorador de Cordial*
6. Implementar el Compilador *PiCO - MAPiCO* tratando de optimizar la generación de variables,
7. Implementar un Garbage Collector dentro de *MAPiCO*,
8. Implementar un cache de objetos,

Recomendaciones

1. Implementar la alternativa de Eliminación de Variables en el Sistema de Restricciones,
2. Cambiar el Sistema de Restricciones a uno mas genérico como *GECODE - Generic Constraint Development Enviroment*,
3. Realizar pruebas funcionales de programas ejecutados desde *Cordial* y no solo desde *MAPiCO*, para comparar tiempos y resultados.
4. Reutilizar la Máquina Abstracta *MAPiCO* para implementar otros cálculos como π o *NTCC*,
5. Implementar un Explorador de Restriciones similar al trabajo de grado *Explorador de Cordial*
6. Implementar el Compilador *PiCO - MAPiCO* tratando de optimizar la generación de variables,
7. Implementar un Garbage Collector dentro de *MAPiCO*,
8. Implementar un cache de objetos,
9. Implementar el soporte de hilos en la Máquina Abstracta *MAPiCO*,

GRACIAS!!!