리눅스에서의 .NET 구현 비교

장익현° 김원영** 최완**
°동국대학교 정보통신공학과
**한국전자통신연구원 온디맨드서비스연구팀
ihjang@dongguk.ac.kr°, {wykim,wchoi}@etri.re.kr**

A Comparison of .NET Implementation in Linux

Ikhyeon Jang°
Dept. of Information Communication Engineering, Dongguk University
Won-Young Kim**, Wan Choi**
Electronics and Telecommunications Research Institute

요 약

마이크로소프트의 닷넷은 언어독립적이고 플랫폼 독립적인 응용을 작성할 수 있도록 해주는 개발환경이다. 웹서비스와 웹응용 분야에서 자바와 경쟁하고 있으나 차세대 인터넷 플랫폼이 될 가능성이 높아 보인다. 닷넷 응용을 리눅스/유닉스 시스템에서 실행할 수 있다는 것은 리눅스/유닉스 시스템에 지금보다 많은 다양한 응용서비스가 제공될 수 있다는 것을 의미한다. 닷넷을 리눅스 상에 구현한 대표적 프로젝트인 Mono와 Portable.NET의 실용성을 비교 시험을 통해 분석하여 보았다.

1. 서 론

마이크로소프트의 .NET은 프로그래밍 언어 독립적이고 플랫폼 독립적인 어플리케이션을 작성할 수 있도록 해주는 강력한 개발환경이다. 웹서비스나 웹 응용 분야에서 자바와 경쟁하고 있으나 마이크로소프트의 영향력을 고려하면 .NET은 차세대 인터넷 플랫폼이 될 가능성이 높아 보인다. 따라서 향후 데스크 탑이나 웹서비스 분야에서 .NET의 영향력은 계속적으로 증가할 것으로 예상된다.

마이크로소프트에 대항하고 있는 공개소프트웨어 진영에서는 리눅스에 대한 지원을 계속하고 있으며, 데스크탑 플랫폼으로도 리눅스가 강력하게 대두되고 있다. 따라서 .NET 기반의 응용서비스를 Linux 운영체제에서도 그대로 실행시킬 수 있으면 Linux 시스템에 지금보다 훨씬 많은 다양한 응용서비스가 제공될 수 있다는 것을 의미한다. 이러한 노력으로 Ximian 주도의 Mono 프로젝트[1]와 DotGNU의 Portable.Net[2] 프로젝트 등이 활발하게 추진되고 있다.

본 연구에서는 리눅스에 닷넷을 구현한 Portable.NET과 Mono의 구현 내용을 살펴보고 두 프로젝트의 구현 내용의 차이점과 성능을 비교하여 실용성을 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 마이크로소프트의 .NET에 대해 간략히 정리하였으며 3장에서는 .NET을 리눅스에 구현하는 과제들을 소개한다. 4장에서는 Mono와 Portable.NET 프로젝트의 기능, 성능에 대한 시험과 분석을 하였으며 5장에서 결론을 맺는다.

2. 닷넷 프레임워크

닷넷 프레임워크는 [그림 1]에서 보는 바와 같이 많은 컴포넌트로 구성된다. 그러나 이들은 다시 C#, CLI와 같은 ECMA/ISO 표준에 포함된 부분과 ASP.NET, ADO.NET, System.Windows.Forms와 같이 표준에 포함되지 않은 부분으로 나눌 수 있다[3]. 닷넷 프레임워크의 컴포넌트에 대한 자세한 설명은 지면상 생략한다.

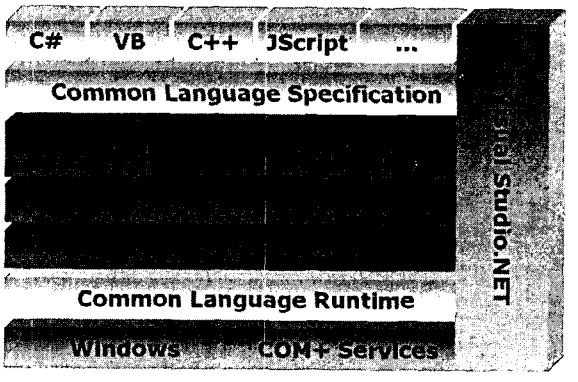

그림 1 닷넷 프레임워크 구조

본 연구는 한국전자통신연구원 "공개 S/W 기반 온디맨드 사무환경 제공 기술 개발"과제의 지원을 받았음.

MS는 C#이 안정되고 독립적인 기술이라는 사실을 고객들에게 확신시키고 닷넷의 확산을 촉진하기 위하여 표준화를 추진하였다. 2000년 8월 MS는 HP, Intel과 공동으로 C#과 CLI를 ECMA 표준안으로 제출하여 2001년 12월 C#과 CLI는 ECMA-334와 ECMA-335 표준으로 채택되었다. [그림 2]는 닷넷에서의 프로그램 컴파일과 실행과정을 보여주고 있다.

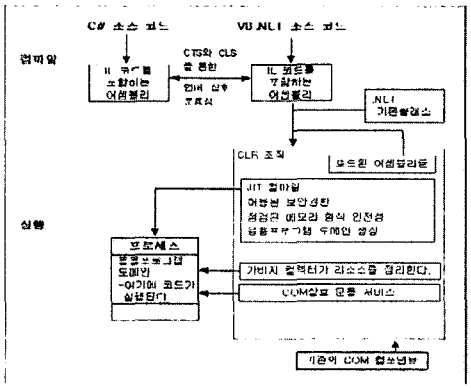

그림 2 닷넷에서의 컴파일과 실행과정

3. 리눅스에서의 닷넷 구현

닷넷 개발환경을 Linux에 구현하기 위한 대표적인 노력으로는 Ximian의 Mono 프로젝트와 DotGNU의 Portable.Net 프로젝트가 있다. 두 프로젝트 모두 닷넷 응용을 Linux 시스템에서 실행할 수 있도록 하는 것이다. 또 MS의 Rotor[4]와 Intel의 OCL[5] 프로젝트도 ISO 표준이 된 C#과 CLI 를 구현하였다. 그러나 뒤의 두 프로젝트는 현재는 큰 의미를 가지지 못하고 있다.

3.1Mono

Gnome을 개발한 공개소스 데스크톱 소프트웨어 회사인 Ximian에서 주도하고 있는 Mono 프로젝트는 초기에는 HP와 Intel사의 도움을 받았으나 최근에 Ximian이 Novell에 인수되면서부터 Novell의 지원을 받고 있으며 2004년 8월 Mono 1.0.1을 발표하였다. 지원되는 운영체제에는 Linux, Windows, FreeBSD, Mac OS X등이 있으며 x86, PowerPC, s390 CPU에서 동작한다.

ECMA 표준 외에도 닷넷 응용을 실행하는데 필요한 라이브러리와 ASP.NET, ADO.NET, VB.NET 등에 대한 개발을 추진하여 높은 완성도를 보이고 있다.

3.2Portable.Net

Portable.Net 프로젝트는 MS의 닷넷 전략에 대항해 닷넷과 호환되는 자유소프트웨어를 개발하여 공급하는 것이다. Portable.Net의 최초 타겟 플랫폼은 GNU/Linux 이나 Windows, NetBSD, FreeBSD, Solaris와 Mac OS X에서도 동작한다. 지원되는 CPU에는 x86, PowerPC, ARM, SPARC, S390, Alpha, IA-64, PA-RISC 등이 있다. 2004년 8월 0.6.8을 발표하였으며 ECMA 표준의 C#과 CLI에 대한 구현에 더하여 System.Windows.Forms를 중심으로 데스크탑 GUI 환경 구축에 많은 노력을 하고 있다.

4. Mono와 Portable.Net 시험

Mono와 Portable.Net 시험은 런타임과 C# 컴파일러, 라이브러리 시험으로 나누어 수행하였으며, Mono와 Portable.Net에서 제공하는 시험 도구를 사용하였다. 두 프로젝트의 비교 시험은 리눅스 시스템에서 하였으며 서로 차이가 나는 구현이 있을 경우에는 Windows XP에서 마이크로소프트 Visual Studio .NET 2003을 사용하여 확인하였다. 시험에 사용한 환경은 다음과 같다.

- Linux 커널 2.4 기반의 Red Hat Linux 9
- Windows XP Professional
- Intel Pentium 4, 2.0 GHz, 512 MB RAM

4.1C#컴파일러

Mono와 Portable.Net 모두 C# 컴파일러의 경우 ECMA 표준을 구현하였으나 Portable.Net은 완성도가 약간 떨어졌다. 예를 들어 [그림 3]은 Mono의 구현과 Portable.Net의 몇 가지 경우에 구현상의 차이를 보이고 있다.

C#에는 boxing과 unboxing이라는 기능이 제공되고 있다. Mono는 MS 닷넷과 동일한 결과를 보이고 있으나 Portable.Net은 약간 다른 결과를 보이고 있다. 이러한 문제는 즉시 보고가 되고 있고 금방 수정할 수 있는 문제이기 때문에 심각한 것은 아니라고 보인다.

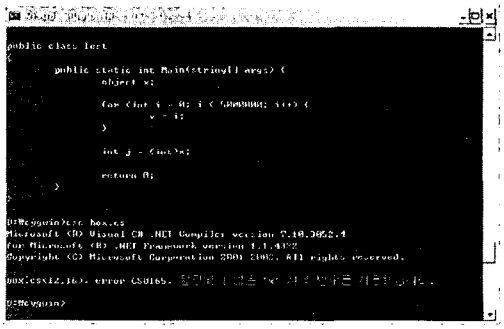

그림 3 Mono와 Portable.Net 비교 시험 예

4.2 런타임

Mono와 Portable.Net이 모두 ECMA 표준을 제대로 구현하여 별 문제를 보이지 않았다. 각 프로젝트에서 제공한 시험을 모두 통과하여 리눅스에서 ECMA 표준을 준용하는 닷넷 응용을 실행하는데는 어려움이 없을 것으로 보인다.

4.3 라이브러리

이 부분은 Mono와 Portable.Net의 구현에서 차이를 보이는 부분이다. Mono는 닷넷을 리눅스에 구현할 때 실제 사용자가 실행하는 응용서비스에 더 높은 관심을 가지고 웹서비스와 DB 처리 지원 등에 비중을 두고 있다. 반면에 Portable.Net은 GUI 데스크탑 환경 지원에 더 높은 비중을 두고 있는 것으로 보인다.

따라서 Mono에서는 닷넷 프레임워크의 라이브러리 외에도 기존의 리눅스 환경에서 제공되고 있던 많은 라이브러리를 Mono 라이브러리로 추가로 제공하고 있다. 그러나 System.Windows.Forms는 아직 충분히 구현하지 못하고 기존의 기존 리눅스 라이브러리를 수정한 GTK# / Qt# 을 제공하여 사용하도록 하고 있다. Portable.Net에서는 System.Windows.Forms를 구현하였으나 나머지 부분은 Mono의 경우만큼 충분히 개발되어 있지 않은 상태이다.

4.4 성능비교

Linpack과 Portable.Net에서 제공하는 벤치마크프로그램인 PNetMark를 사용하여 Portable.Net과 Mono의 성능을 비교하였으며 결과는 표 1과 표 2로 나타내었다. 다. Mono는 컴파일러(mcs)버전과 해석기(mint) 버전 두 가지를 포함하였다.

시험은 벤치마크 프로그램을 각 프로젝트의 컴파일러를 사용하여 컴파일하고 해당 프로젝트의 런타임에서 실행한 결과를 나타낸 것이다.

표 1 PNetMark 시험 결과의 일부

	loop	string	method
Portable.Net	33596	1603	4090
Mono(mcs)	63289	1681	27289
Mono(mint)	2291	1273	1048

[표 1]의 결과를 보면 Portable.Net의 벤치마크 프로그램을 사용하였으나 Mono가 더 좋은 성능을 보여주고 있다. 특히 Mono의 컴파일러 버전은 x86에 대한 JIT 엔진을 포함하기 때문인지는 분명하지 않으나 해석기를 사용한 것과 많은 차이를 보여주고 있다.

표 2 Linpack 시험 결과 (n=500)

	Mflops/s	Time
Portable.Net	11.17	7.52
Mono(mcs)	138.22	0.61
Mono(mint)	5.65	14.85

Linpack 시험에서도 Mono의 컴파일러 버전이 가장 좋은 결과를 보여주고 있다.. Portable.Net은 Mono의 해석기에 비해서는 좋은 결과를 보여주고 있다.

5. 결론

본 연구에서는 마이크로소프트의 닷넷을 리눅스에 구현한 Mono와 Portable.Net 프로젝트의 구현 내용을 살펴보고 그들을 비교 시험하여 보았다.

전체적으로 Mono의 구현정도는 Portable.Net의 구현보다 높은 완성도를 보이고 있다. Mono의 경우 웹서비스와 DB 지원 부분도 상당한 정도까지 와 있는 상태이다. Portable.Net의 경우에는 아직 충분한 완성도를 보이지 못하고 있으나 GUI 부분에서는 Mono보다 높은 정도로 구현하고 있는 상태이다.

성능 비교에서도 Mono가 Portable.Net보다 상대적으로 좋은 것으로 나타나고 있다.

참 고 문 헌

[1] Mono, http://www.go-mono.com

[2] Portable.Net, http://www.dotgnu.org/pnet.html

[3] J.Richter, Applied Microsoft .NET Framework programming, Microsoft Press, 2002

[4] Rotor, http://www.research.microsoft.com/programs/europe/rotor/default.aspx

[5] OCL, http://sourceforge.net/projects/ocl