



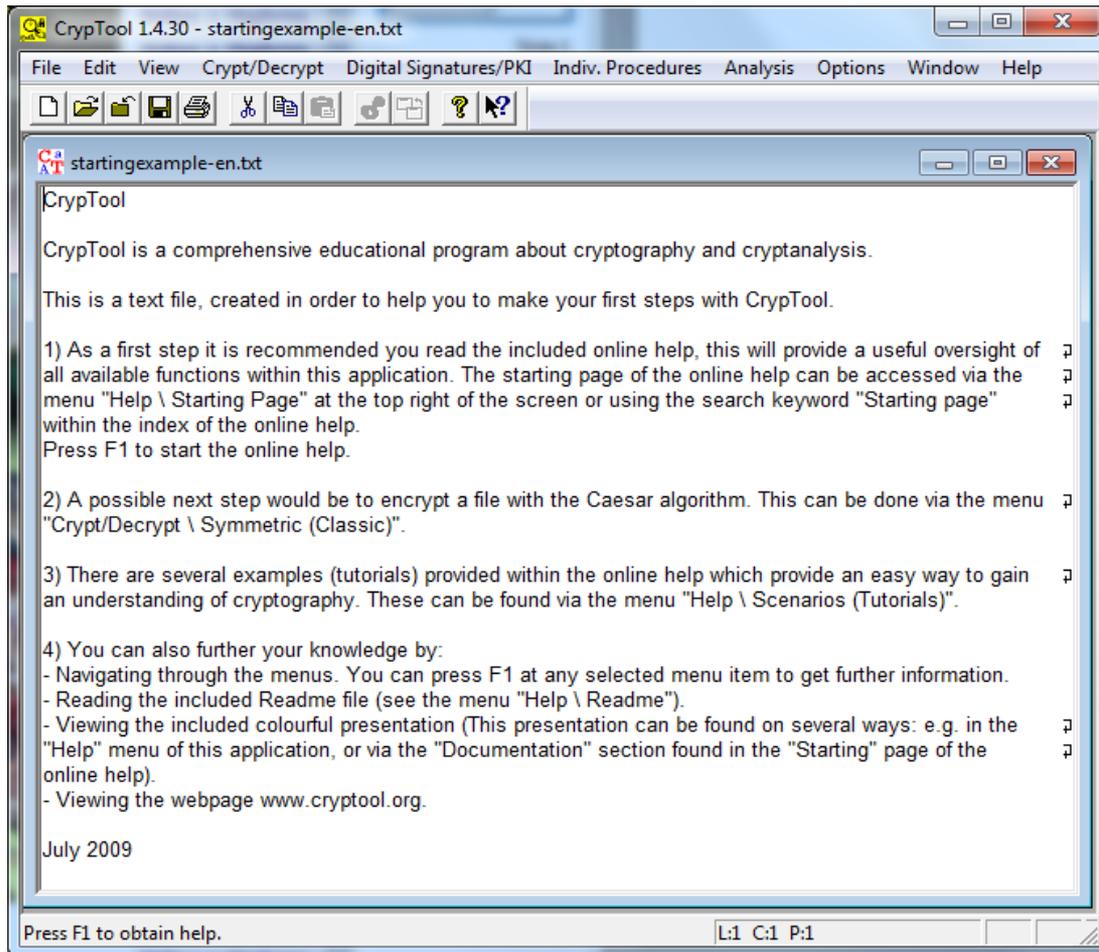
The golden age of
hacking

Parallel Computing
GPGPU and
password/hash/crypto
attacks

CrypTool

Cryptography for the masses

Recommended!



CrypTool (Freeware)

Crypt methods

Analysis

Visualisations

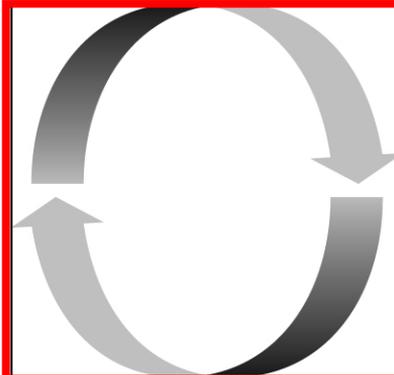
Etc. etc....

<http://www.cryptool.org/>

Password attacks I

http://en.wikipedia.org/wiki/Password_cracking

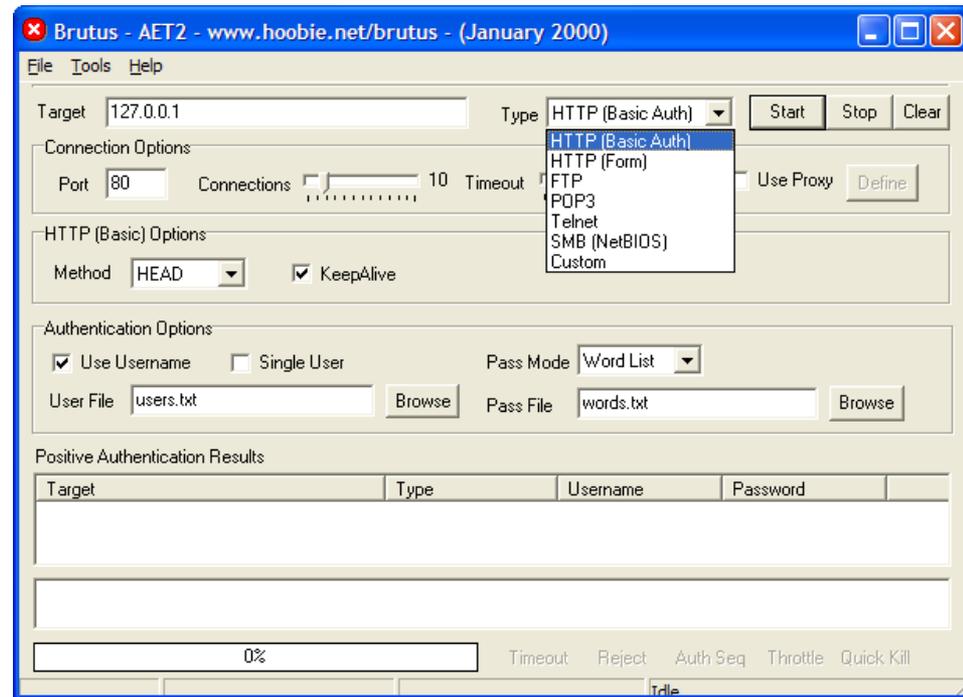
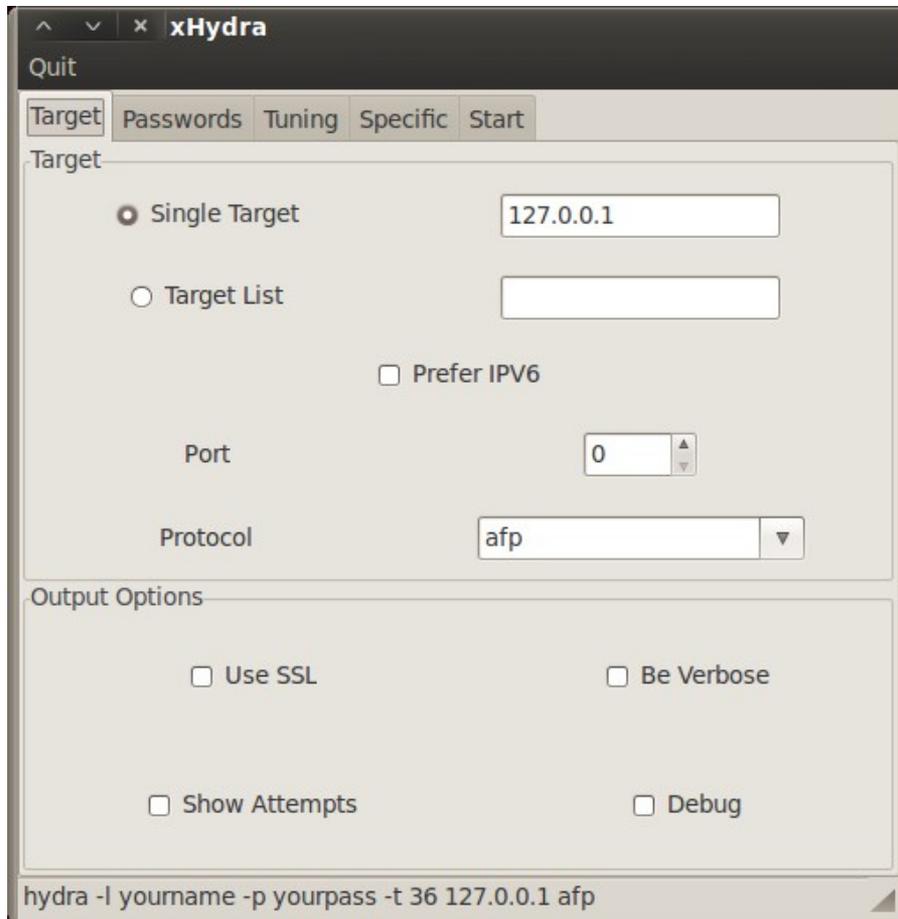
- Topic is already rather known by most of you!
 - At least the cracking part
- Default built in passwords in systems
- Network login tools (**password guessing**)
 - Brutus
 - THC Hydra
 - Supported services includes: TELNET, FTP, HTTP, HTTPS, HTTP-PROXY, SMB, SMBNT, MS-SQL, MYSQL, REXEC, RSH, RLOGIN, CVS, SNMP, SMTP-AUTH, SOCKS5, VNC, POP3, IMAP, NNTP, PCNFS, ICQ, SAP/R3, LDAP, PostgreSQL, Teamspeak, Cisco auth, Cisco enable, and Cisco AAA.
 - Reset attack
 - Account lockout (DoS)
- Passwords are either
 - Hashed or encrypted
 - Flaw in algorithm?
 - Entropy?



- Create a password guess
- Encrypt the guess
- Compare encrypted guess with encrypted value from the stolen password file
- If match, you've got the password!
Else, loop back to the top.

Network login tools

- THC (The Hackers Choice) Hydra Xhydra, #3
- Brutus, old, #10 on <http://sectools.org>



Entropy (password entropy)

$$2^{6,56}=95$$

- Measuring password strength (disorder)
- For a completely random password, each character is worth approximately 6.56 bits
- With a user-chosen, first sign give 4 bits, characters 2-8 gives 2 bits, characters 9-20 gives 1.5 and 21-... provides 1 bit per character
- Entropy (bits) table for various lengths of passwords

Length (chars)	8	20	63
User-choosen (freely choosen)	18	36	79
User-choosen (according to rules)	30	42	85
Random	52	131	413

- The number of variations for a password is $2^{(\text{number of bits})}$
- However $(\text{number of possible characters})^{(\text{number of characters})}$ is basically flawed because a short passwords can be complex and long passwords can be of an easily guessable character
- http://en.wikipedia.org/wiki/Password_strength

Password attacks II

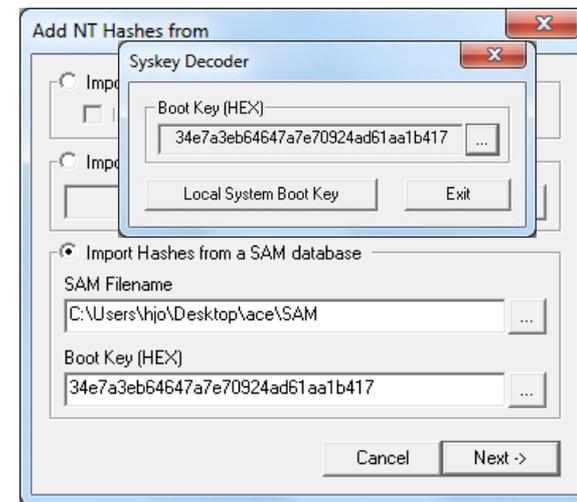
- Dictionary → Wordlists
 - Google search "wordlist compilation"
- Wordlists vs brute-force
 - keyspace_password.xls
 - Hybrid attacks (permutations)
 - Distributed attacks - botnets
- The best free tools
 - Cain & Abel (#1)
 - Swiss army knife
 - John the ripper (#2)
 - Multi platform, permutation
- SAM registry/DB file attacks
 - Online
 - Cain, fgdump, etc.
 - Offline
 - Bootdisks
 - Ophcrack (Vista/7)
 - Reset



Offline extraction of credentials

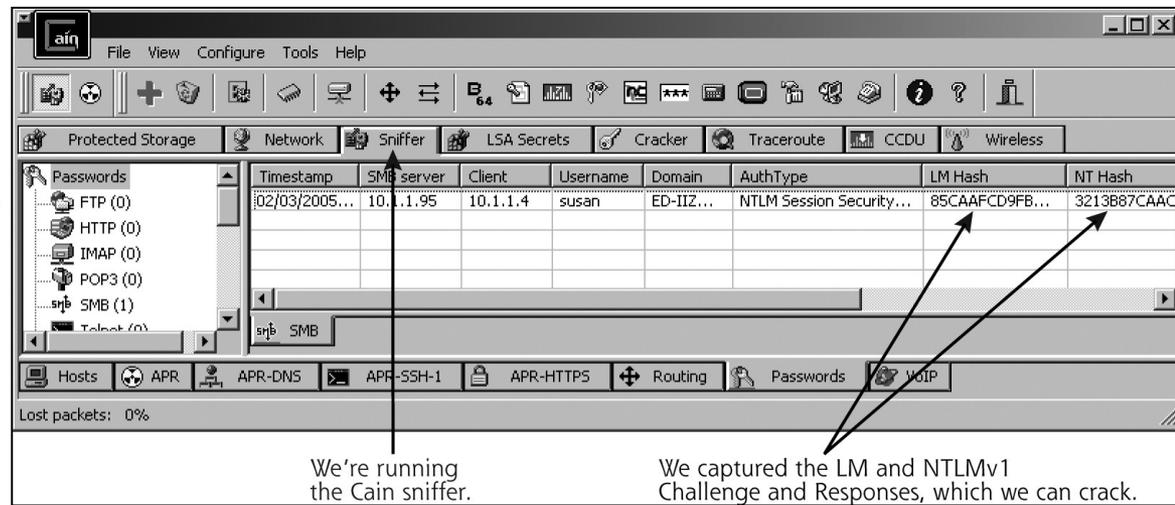
- Hash encryption in article “Syskey and SAM” at: <http://moyix.blogspot.com/2008/02/syskey-and-sam.html>
- Creddump (Python scripts)
 - LM and NT hashes (Syskey protected – 128 bits)
 - Cached domain credentials and LSA secrets
 - <http://code.google.com/p/creddump/>
- Other tools
 - Cain – from Forensic 1, lab 4.8
 - Add NT Hashes, Syskey Decoder (System), ...
 - SAMInside
 - Bkhive (dump syskey), Samdump2 etc.
- Tutorials: IronGeek
<http://www.irongeek.com/i.php?page=security/cracking-windows-vista-xp-2000-nt-passwords-via-sam-and-syskey-with-cain-oplcrack-saminside-bkhive-etc>

syskey.exe



Cain and Windows hashes

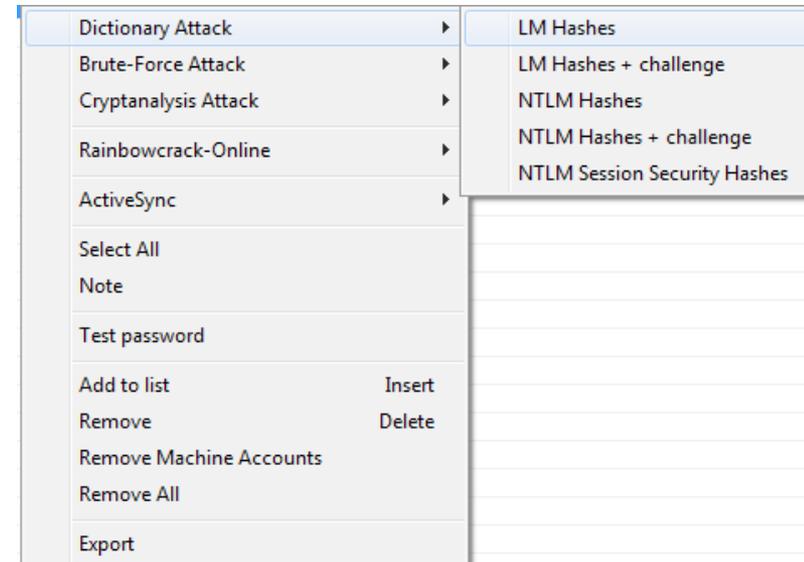
- Microsoft LM or LanMan
 - The weak uppercase fixed 14 digit chopped ...



We're running the Cain sniffer.

We captured the LM and NTLMv1 Challenge and Responses, which we can crack.

- Windows NT(LM) hash
 - Much stronger than LM (MD4 x 3)
- Network sniffed authentication packets
 - LM challenge-response
 - NTLMv1 challenge-response
 - Stronger than LM challenge
 - NTLMv2 challenge-response
 - Stronger than NTLMv1
 - MS-Kerberos5 pre auth
 - The MS version used in most AD networks
- CHR page 137 and 386 – 396



Cain and Windows hashes cont.

Cain can determine which passwords are seven characters or less by observation, because encrypted padding is always **AAD3B43...** with no salts.

The screenshot shows the 'Cracker' window in Cain software. The table below lists users and their hashes. Annotations include: a red box around 'AAD3B43...' in the text above; a red box around the 'NT Password' column header; a red box around the 'NT Password' and 'LM Hash' columns for the 'susan' user; arrows pointing from the 'User Name' column to the text 'User IDs dumped from the SAM database'; arrows pointing from the 'NT Password' and 'LM Hash' columns to the text 'Cracked LM and NT password representations (the question marks indicate that the upper seven characters of an LM hash haven't yet been cracked)'; and arrows pointing from the 'LM Hash' and 'NT Hash' columns to the text 'The original LM and NT representations dumped from the SAM'. A large bracket on the left side of the interface is labeled 'Different cracking tools for numerous different password representations'. The URL 'http://www.oxid.it' is visible at the bottom left.

User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	Challar
Administrator				834B5B2B8BFB4C7A2BCF4FF...	505BB022EFF2...	
alice	NUGGET	*	nugget	5EFB4E796578082BAAD3B43...	82FA5017A8F2...	
fred	LETMEIN	*	letmein	5D567324BA3CCE8AAD3B43...	BECEDB42EC3C...	
Guest	* empty *	*	* empty *	AAD3B435B5140E5EAAD3B43...	31D6CFE0D16A...	
Robert	* empty *	*	* empty *	AAD3B435B5140E5EAAD3B43...	31D6CFE0D16A...	
susan	PASSWOR??...			E52CAC67419A9A2236077A7...	5F946A12C3EB...	

Different cracking tools for numerous different password representations

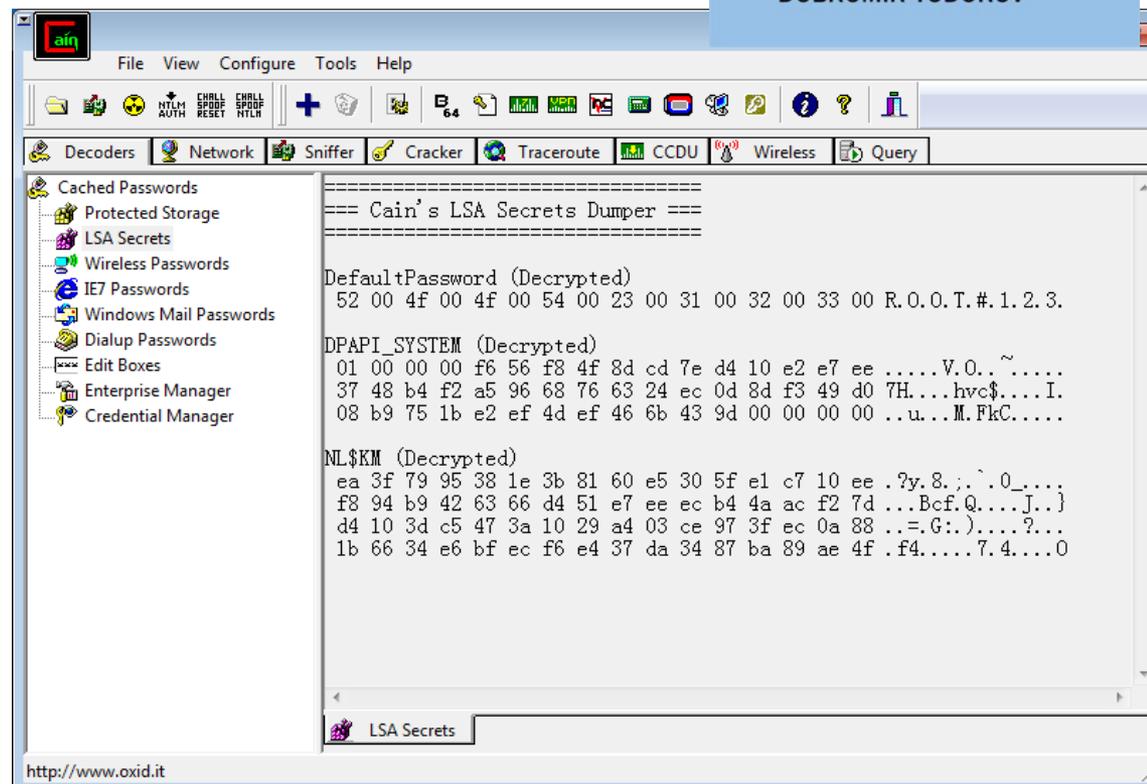
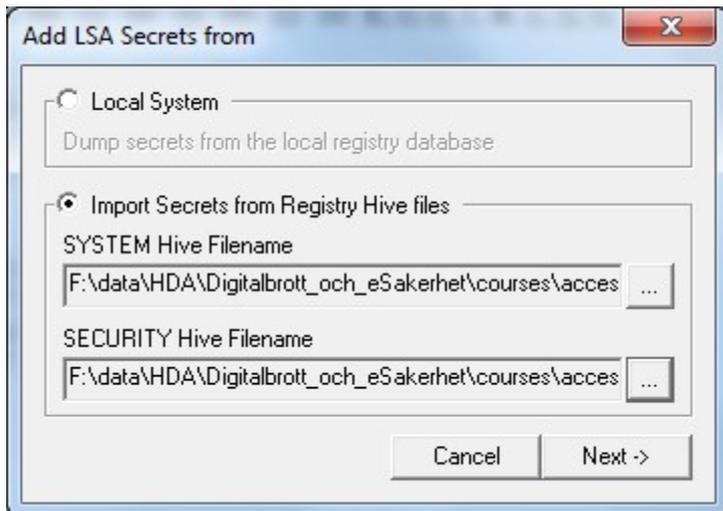
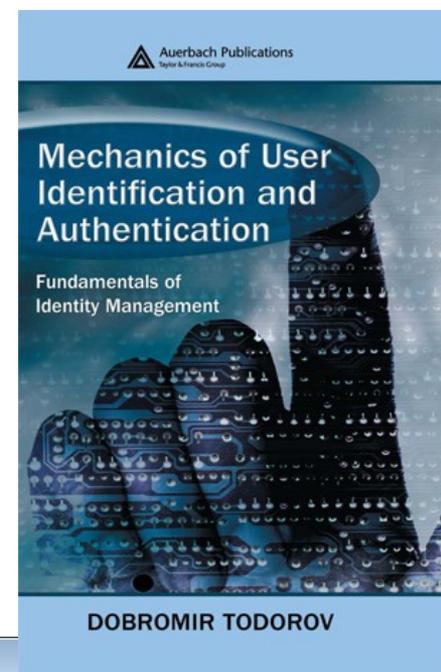
User IDs dumped from the SAM database

Cracked LM and NT password representations (the question marks indicate that the upper seven characters of an LM hash haven't yet been cracked)

The original LM and NT representations dumped from the SAM

Cain LSA secrets

- Decrypting Local Security Authority (LSA) secrets you may find
 - DefaultPassword – used if auto-login is enabled
 - NL\$KM – secret key used to encrypt cached domain passwords
 - Various service account secrets, \$MACHINE.ACC, etc...
- DPAPI_SYSTEM is a legacy backup key that is used to recover DPAPI (Data Protection Application Programming Interface) data
- Very good book describing algorithms



Unix passwd and shadow files

- Salt (a small extension, 2-8 byte) often used to complicate rainbow attacks
hash = OWF(password + salt) - Unix use salt, Windows does not
http://en.wikipedia.org/wiki/Salt_%28cryptography%29

“\$1\$”=MD5, “\$5\$”=sha-256, “\$6\$”=sha-512, (man shadow, crypt (3))

The /etc/passwd file holds user account information, including login name, User ID number, GroupID number, user comment (called the GECOS field), home directory and shell.

```
root@test root]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./:/sbin/nologin
rpm:x:37:37:./var/lib/rpm:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
nscd:x:28:28:NSCD Daemon:./:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:./:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
pcap:x:77:77:./var/arpwatch:/sbin/nologin
mailnull:x:47:47:./var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:./var/spool/mqueue:/sbin/nologin
dbus:x:81:81:System message bus:./:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
ntp:x:38:38:./etc/ntp:/sbin/nologin
gdm:x:42:42:./var/gdm:/sbin/nologin
alice:x:502:502:./home/alice:/bin/bash
fred:x:503:503:./home/fred:/bin/bash
susan:x:504:504:./home/susan:/bin/bash
robert:x:505:505:./home/robert:/bin/bash
root@test root]#
```

Here are the user accounts that aren't associated with the operating system but are instead assigned to people (uid > 500).

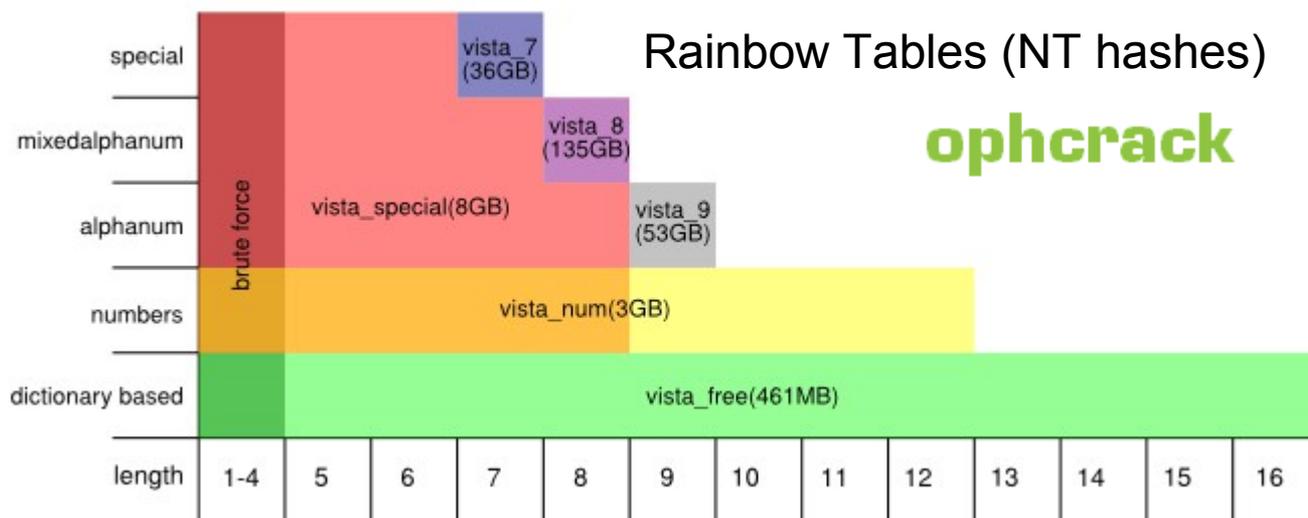
```
root@test root]# cat /etc/shadow
root:$1$JwHcdy9$VTILqgawBy42T0dGtVwh0:12662:0:99999:7:::
bin*:12662:0:99999:7:::
daemon*:12662:0:99999:7:::
adm*:12662:0:99999:7:::
lp*:12662:0:99999:7:::
sync*:12662:0:99999:7:::
shutdown*:12662:0:99999:7:::
halt*:12662:0:99999:7:::
mail*:12662:0:99999:7:::
news*:12662:0:99999:7:::
uucp*:12662:0:99999:7:::
operator*:12662:0:99999:7:::
games*:12662:0:99999:7:::
gopher*:12662:0:99999:7:::
ftp*:12662:0:99999:7:::
nobody*:12662:0:99999:7:::
rpm:!:12662:0:99999:7:::
vcsa:!:12662:0:99999:7:::
nscd:!:12662:0:99999:7:::
sshd:!:12662:0:99999:7:::
rpc:!:12662:0:99999:7:::
rpcuser:!:12662:0:99999:7:::
nfsnobody:!:12662:0:99999:7:::
pcap:!:12662:0:99999:7:::
mailnull:!:12662:0:99999:7:::
smmsp:!:12662:0:99999:7:::
dbus:!:12662:0:99999:7:::
xfs:!:12662:0:99999:7:::
ntp:!:12662:0:99999:7:::
gdm:!:12662:0:99999:7:::
alice:$1$a36//8We$nQszd1GSN1kAL2r2ctNIL/:12845:0:99999:7:::
fred:$1$8E1NLd0g$GDHdovsqcPrju3WuQHv2v/:12845:0:99999:7:::
susan:$1$jYvYCP1Z$YzjdcheL8ujvE7yIQApgA/:12845:0:99999:7:::
robert:!:12845:0:99999:7:::
root@test root]#
```

Here are the encrypted passwords.

Uh-oh! Robert has a blank password!

Rainbow tables

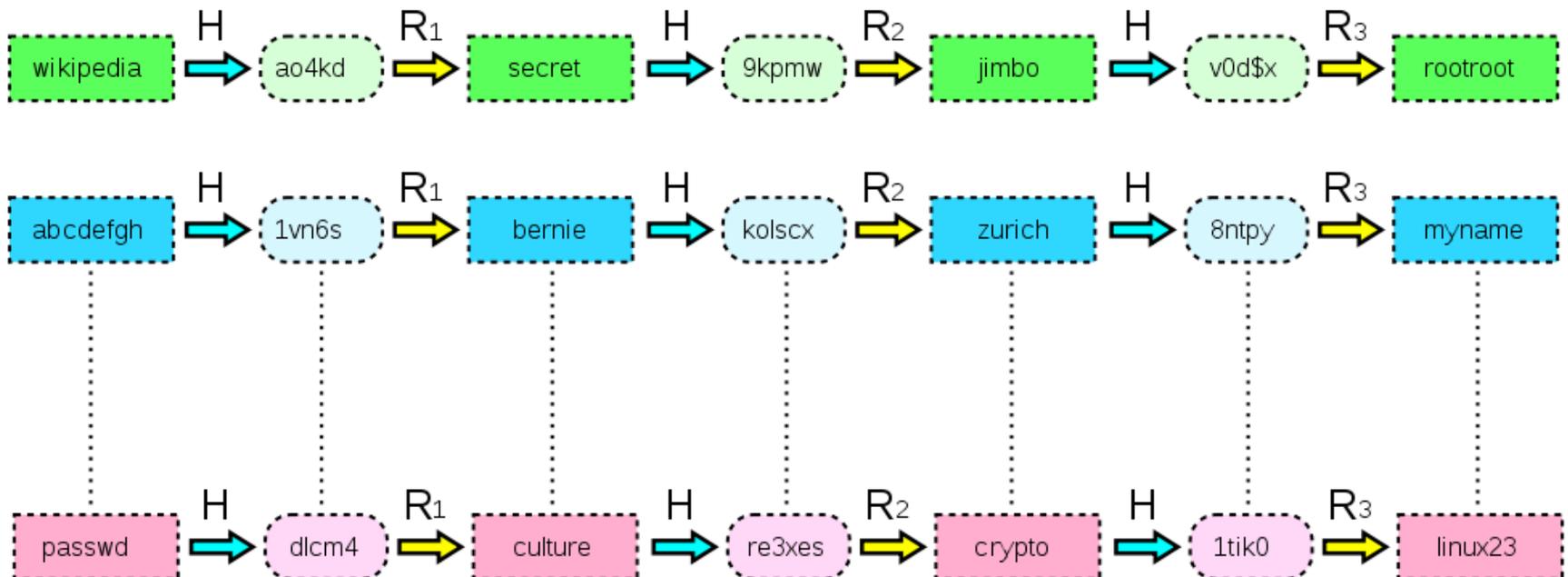
- A refinement (by Philippe Oechslin) of an earlier, simpler algorithm by Martin Hellman that used the inversion of hashes by looking up precomputed hash chains – **note!** not as in: $h(h(h(\text{password})))$
 - Cryptoanalytic time-memory trade-off (fast attack but use more memory)
 - Reduction functions - only the first and last password of a chain stored in table
 - A hit means that chain contains hash - not 100% guarantee to crack password
- Ophcrack – Multi platform/core, special tools as live CD etc.
- Free Rainbow Tables – Windows (src), multi core support, slow updates
- RainbowCrack – Multi platform/core and GPU (CUDA) accelerated
- Cain – Winrtgen, can use other tables as well



Rainbow table example

http://en.wikipedia.org/wiki/Rainbow_table

- A simplified rainbow table with 3 reduction functions
- Chain length is usually up to around 3 – 4 thousand and number of rows is usually around 40 million when expanded fully - if needed
- Functions
 - H = hash function, R = reduction function

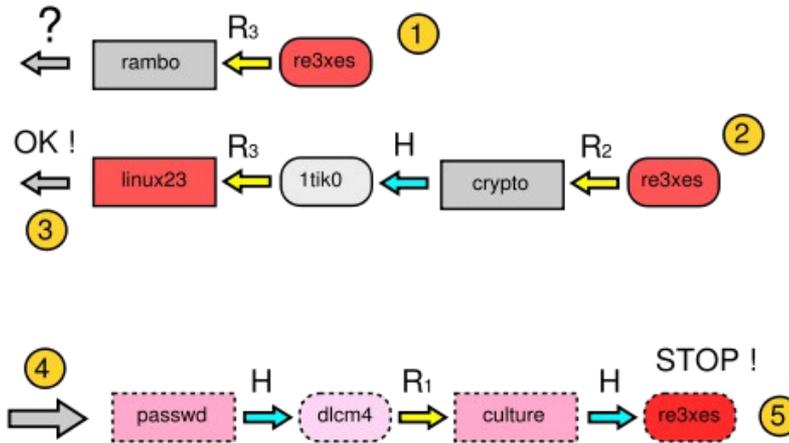
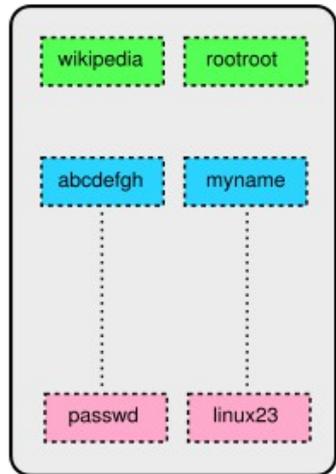


Rainbow table example cont.

http://en.wikipedia.org/wiki/Rainbow_table

We have a hash (re3xes) and we want to find the password that produced that hash

Table with only first and last password stored in chain for every row



Functions
H = hash
R = reduction

1. Starting from the hash ("re3xes"), one computes the last reduction used in the table and checks whether the password appears in the last column of the table (step 1).
2. If the test fails (rambo doesn't appear in the table), one computes a chain with the two last reductions (these two reductions are represented at step 2)
Note: If this new test fails again, one continues with 3 reductions, 4 reductions, etc. until the password is found. If no chain contains the password, then the attack has failed.
3. If this test (step1) is positive (as in step 3, linux23 appears at the end of the chain and in the table), the password is retrieved at the beginning of the chain that produces linux23. Here we find passwd at the beginning of the corresponding chain stored in the table.
4. At this point (step 4), one generates a chain and compares at each iteration the hash with the target hash. In this case the test is valid and we find the hash re3xes in the chain (step 5). The current password (culture) is the one that produced the whole chain : the attack was successful!

Cain Winrtgen v2.9

- Rainbow Table properties

Rainbow Table properties

Hash: Min Len: Max Len: Index: Chain Len: Chain Count: N* of tables:

Charset:

Table properties
Key space: 2901713047668 keys
Disk space: 23,84 GB (610,35 MB each table)
Success probability: 0.843170 (84.32%)

Benchmark
Hash speed: 4999999 hash/sec
Step speed: 975419 step/sec
Table precomputation time: 1.61374 days
Total precomputation time: 64.5497 days
Max cryptanalysis time: 3.95044 minutes

Password attack defense

- Strong password policy
- User awareness
- Password filter
 - Force the use of strong passwords
- Do password-cracking tests
- Protect the hashed password files
- Get rid of LM hashes
 - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
NoLMHash = 1
- Two-factor (or even three) authentication - strong authentication
 - http://en.wikipedia.org/wiki/Two-factor_authentication
 - Something you have (token), you know (passwd), you is or does (fingerprint)
 - SecurID server connected to AD etc.
 - Pin code (number) is generated which matches the RSA SecurID token
 - Token is synchronized with SecurID server
 - Login with both pin code and pass code (displayed in token for limited time)
 - <http://en.wikipedia.org/wiki/SecurID>



Password reuse

<http://xkcd.com/792/>

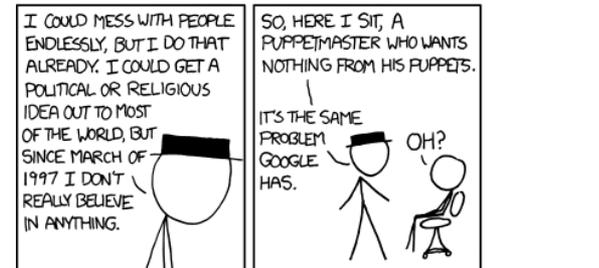
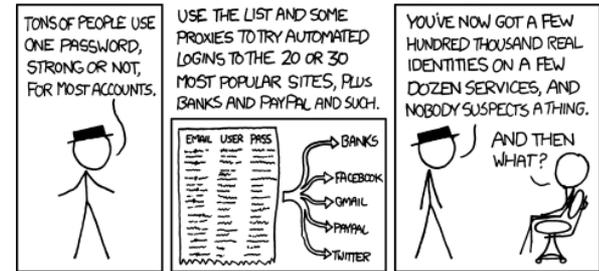
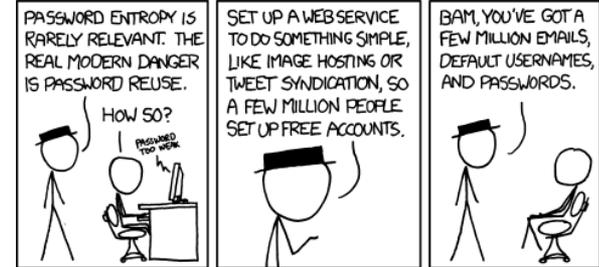
<http://www.xkcd.com/1286/>

HACKERS RECENTLY LEAKED 153 MILLION ADOBE USER EMAILS, ENCRYPTED PASSWORDS, AND PASSWORD HINTS.

ADOBE ENCRYPTED THE PASSWORDS IMPROPERLY, MISUSING BLOCK-MODE 3DES. THE RESULT IS SOMETHING WONDERFUL:

USER	PASSWORD	HINT
4e18acc1ab27a2d6		WEATHER VANE SWORD
4e18acc1ab27a2d6		
4e18acc1ab27a2d6	a0a2876eb1ea1fca	NAME 1
8babbb6279e06eb6d		DUH
8babbb6279e06eb6d	a0a2876eb1ea1fca	
8babbb6279e06eb6d	85e9da81a8a78adc	57
4e18acc1ab27a2d6		FAVORITE OF 12 APOSTLES
1ab29ac86dab6e5ca	7a2d6a0a2876eb1e	WITH YOUR OWN HAND YOU HAVE DONE ALL THIS
a1f9b2b6299e7a2b	e0dec1e6ab797397	SEXY EARLOBES
a1f9b2b6299e7a2b	617ab027727ad85	BEST TOS EPISODE
39738b7adb0b8af7	617ab027727ad85	SUGARLAND
1ab29ac86dab6e5ca		NAME + JERSEY #
877ab7889d3862b1		ALPHA
877ab7889d3862b1		
877ab7889d3862b1		
877ab7889d3862b1		OBVIOUS
877ab7889d3862b1		MICHAEL JACKSON
38a7c9279c0deb44	9dca1d79d4dec6d5	
38a7c9279c0deb44	9dca1d79d4dec6d5	HE DID THE MASH, HE DID THE
38a7c9279c0deb44		PURLAINED
a8ae5745a2b7a77a	9dca1d79d4dec6d5	FAV. LATER-3 POKEMON

THE GREATEST CROSSWORD PUZZLE IN THE HISTORY OF THE WORLD





Death of the web user/password?

<http://en.wikipedia.org/wiki/OpenID>

Authenticate with Google, FB, etc.

- OpenID is a decentralized single sign-on system
 - Authenticate once and gain access to the resources of multiple software systems
- Builds on digital identity from a OpenID provider
 - URL or XRI (eXtensible Resource Identifier)
 - Log in with for example: <http://alice.myopenid.com>
 - Web site will check with OpenID provider if valid (or my own domain which redirects)
- There are two modes in which the relying party can communicate with the identity provider:
 - `checkid_immediate`, which is machine-oriented and in which the relying party requests that the provider not interact with the user. All communication is relayed through the user's browser, but presumably without the user's knowledge;
 - `checkid_setup`, in which the user communicates with the provider server directly using the very same web browser used to access the relying party site.
- OpenID does not provide its own form of authentication, but if an identity provider uses strong authentication, OpenID can be used for secure transactions such as banking and e-commerce (a token or other hardware is needed)
- More reading
 - <http://stacktrace.se/2007/10/04/openid-en-introduktion/>
 - <http://www.intertwingly.net/blog/2007/01/03/OpenID-for-non-SuperUsers>
 - <http://www.idg.se/2.1085/1.338358/google-opnar-for-openid> (2010-09-08)

Death of the web user/password?

YubiKey Core features

- two-factor authentication with one-time passwords
 - Works instantly, no need to re-type pass codes from a device
 - Works on Windows, Mac, Linux, iPad, Firefox, Chrome, etc
 - Identified as a USB-keyboard, no client software or drivers needed
 - Minimized size; 2 mm thin, 3 grams
 - Practically indestructible; waterproof, crush safe, no battery
 - Integration within minutes with free and open source server software
 - Two slots for multiple configurations: OATH*, Challenge-Response etc.
 - Also available with NFC (NEO) and minimized form factor (Nano)
 - Lowest total cost of ownership for strong two-factor authentication
- * OATH (Open Authentication) open standard from VeriSign



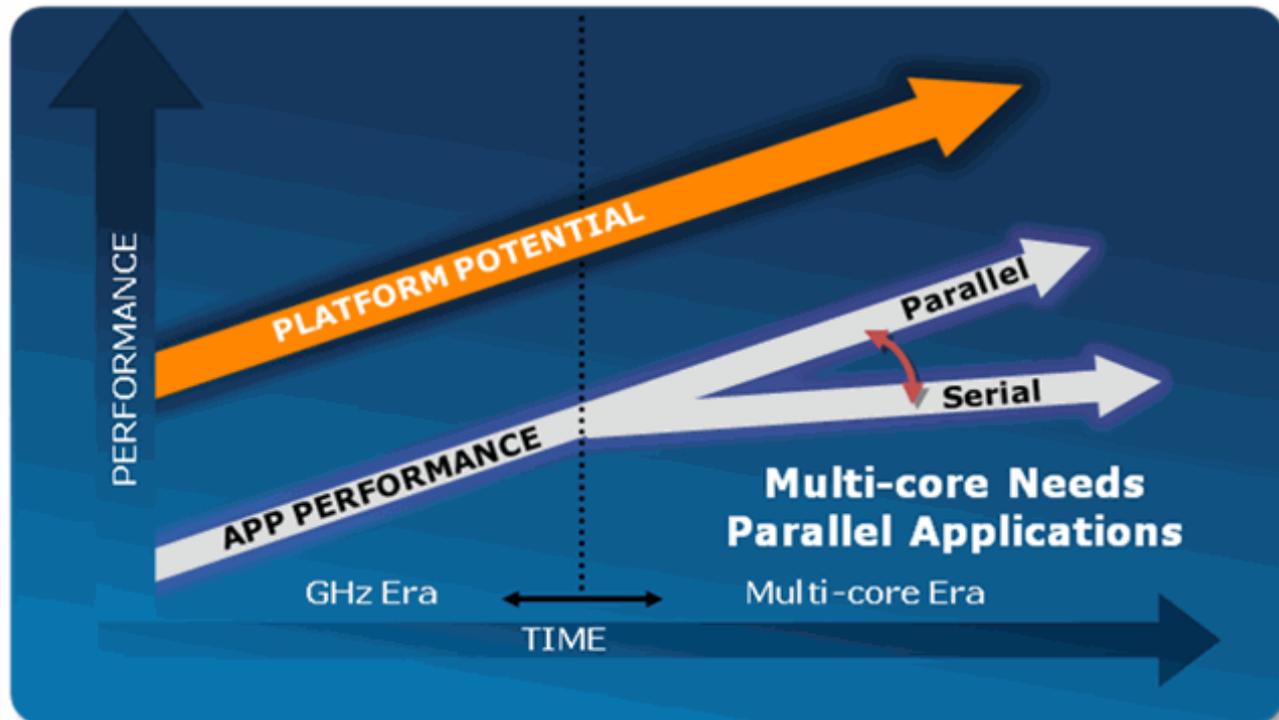
How it works

With a simple touch of the gold disc, the YubiKey sends a One Time Password (OTP) as if it was typed in from a keyboard. The unique passcode is verified by a YubiKey compliant application.



Parallel computing 1

- Good intro to parallel computing
 - https://computing.llnl.gov/tutorials/parallel_comp/
- FASTRA II = **12 TFLOPS**, 13x GPU (6 NVIDIA GTX295 dual-GPU cards and one GTX275 single-GPU card) – GT200, 6000 € (2009)
 - http://en.wikipedia.org/wiki/Fastra_II
- **46 TFLOP** - 2008-06, 14 mil. SEK - <http://www.hpc2n.umu.se/resources/akka/>

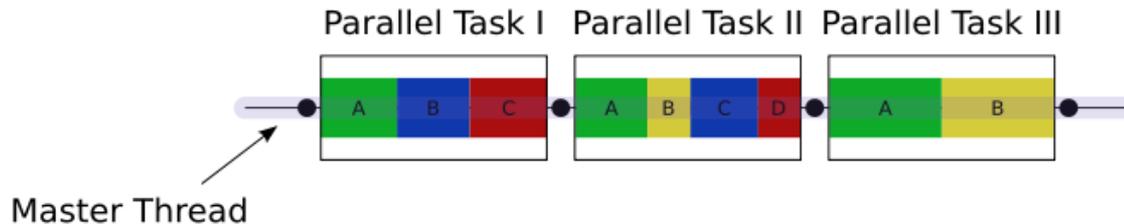


Parallelism

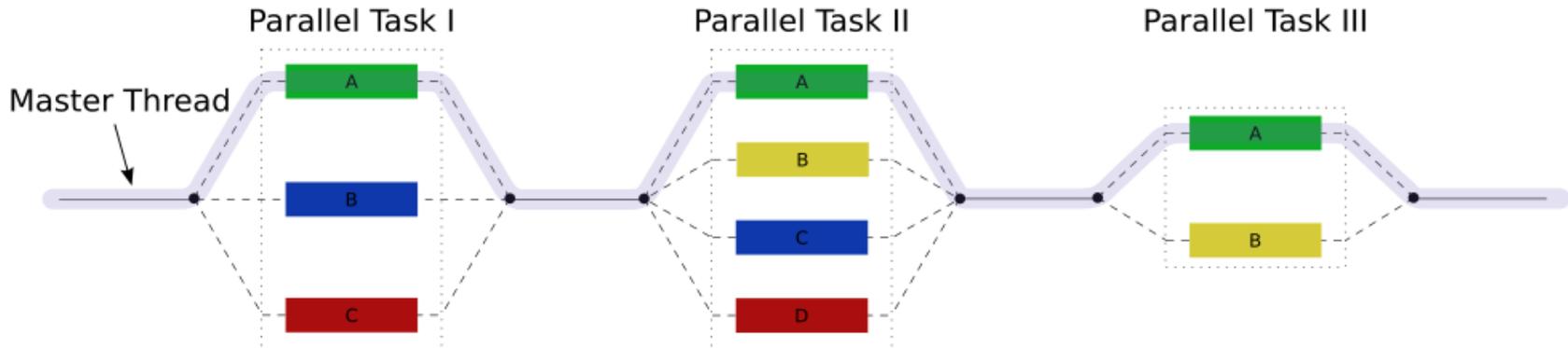
- Data parallelism
 - http://en.wikipedia.org/wiki/Data_parallelism
- Task parallelism
 - http://en.wikipedia.org/wiki/Task_parallelism

Software threads vs.
hardware threads

Data parallelism



Task parallelism



Parallel computing 2

A New Era of Processor Performance

Single-Core Era

Constrained by:

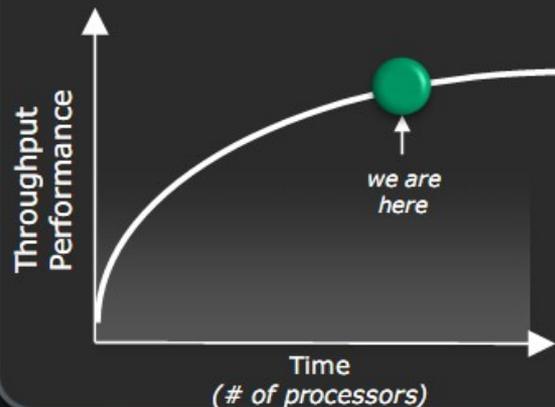
- ✗ Power
- ✗ Complexity



Multi-Core Era

Constrained by:

- ✗ Power
- ✗ Parallel SW availability
- ✗ Scalability



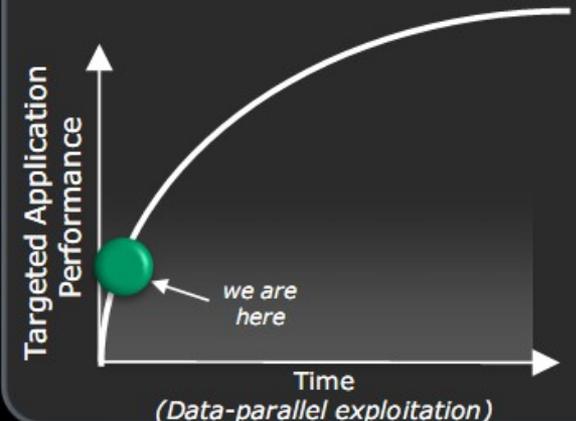
Heterogeneous Systems Era

Enabled by:

- ✓ Abundant data parallelism
- ✓ Power efficient GPUs

Constrained by:

- ✗ Programming models

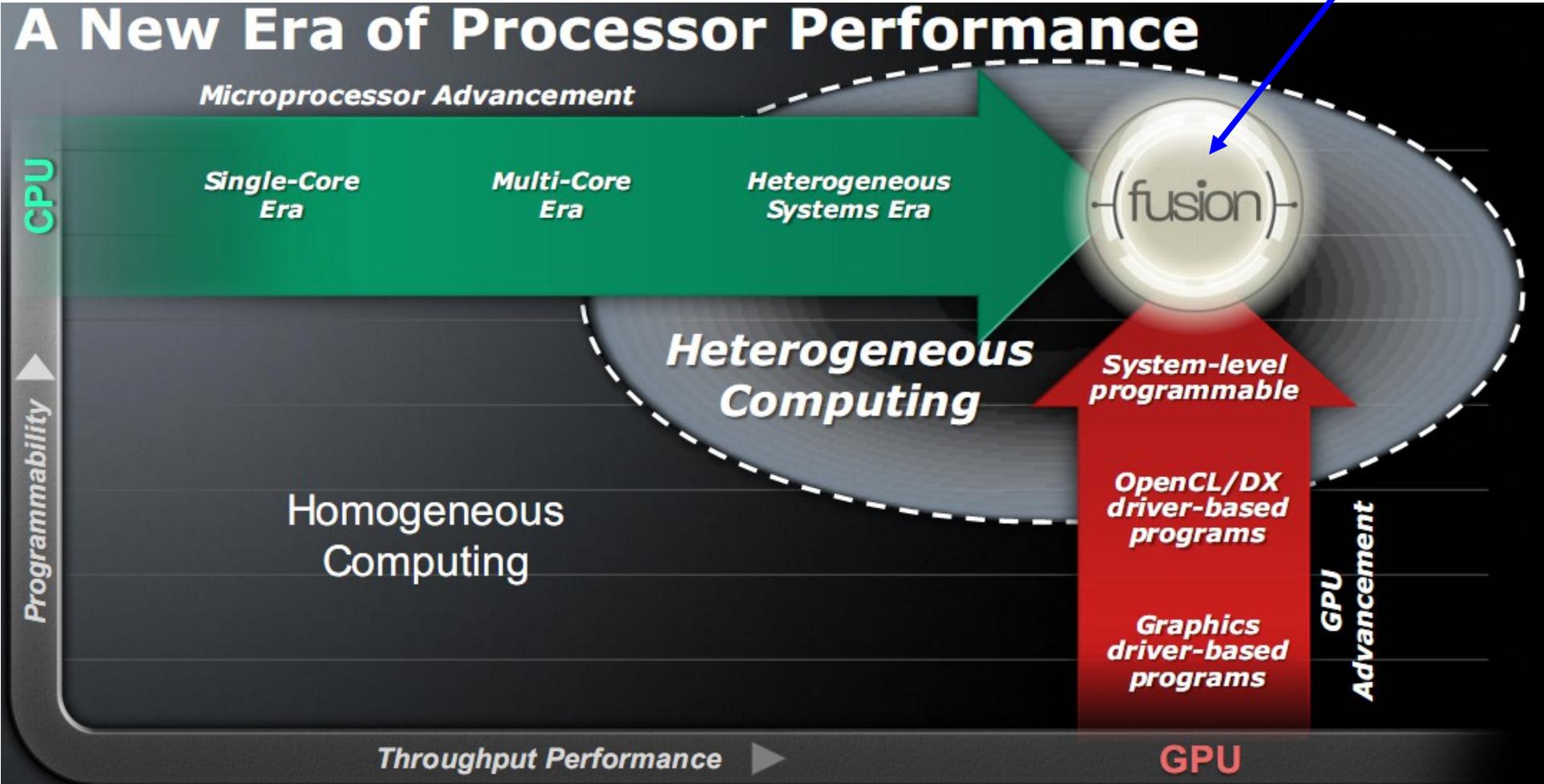


Just now 6, 8, 10 and 12 "many core" CPUs can be bought

Parallel computing 3

Radeon HD 6450 in room 348 have 160 SP

AMD Llano
X4 CPU, 400 SP



Fusion → APU (Accelerated Processing Unit), CPU + GPU

Tools for multicore development

- Writing native threads is now considered a relic from the early days of parallelism
- Over 40% of all programmers now work on data parallel applications according to ZDNet (2011)
 - A popular solution then were Intel Threading Building Blocks (& OpenMP)
- Most of the new libraries and extensions builds on the work-stealing algorithm pioneered by the Cilk project (now Cilk++)
 - <http://en.wikipedia.org/wiki/Cilk> - <http://www.cilk.com/>
- Work-stealing algorithm in short
 - Each worker thread maintains runnable tasks in its own scheduling queue
 - When a worker thread has no local tasks to run, it attempts to take("steal") a task from another randomly chosen worker thread, using FIFO (oldest first) rule
 - When a worker thread encounters a join operation, it processes other tasks, if available, until the target task is noticed to have completed
 - When a worker thread has no work and fails to steal any from others, it backs off (via yield, sleep, and/or priority adjustment)

(Open Multi-Processing) API

- Supports multi-platform shared memory multiprocessing programming in C/C++/Fortran with both task and data parallelism via preprocessor pragma directives
 - Threads usually joins (barrier) with the master thread which got id=0
 - Makes it simple to add multi-core support into existing programs
 - MSDN Magazine article: "Reap the Benefits of Multithreading without All the Work"

```
// compile with: /openmp to enable OpenMP 2.0 language extensions
// http://msdn.microsoft.com/en-us/library/tt15eb9t.aspx
#include <stdio.h>
#include <omp.h>
int main()
{
    printf("OMP Max Threads: %d\n", omp_get_max_threads());
    printf("OMP Num Cores: %d\n", omp_get_num_procs());

    #pragma omp parallel sections num_threads(2)
    {
        #pragma omp section
        printf_s("Hello from thread %d\n", omp_get_thread_num());
        #pragma omp section
        printf_s("Hello from thread %d\n", omp_get_thread_num());
    }
    // Implicit barrier synchronize the threads
}
```

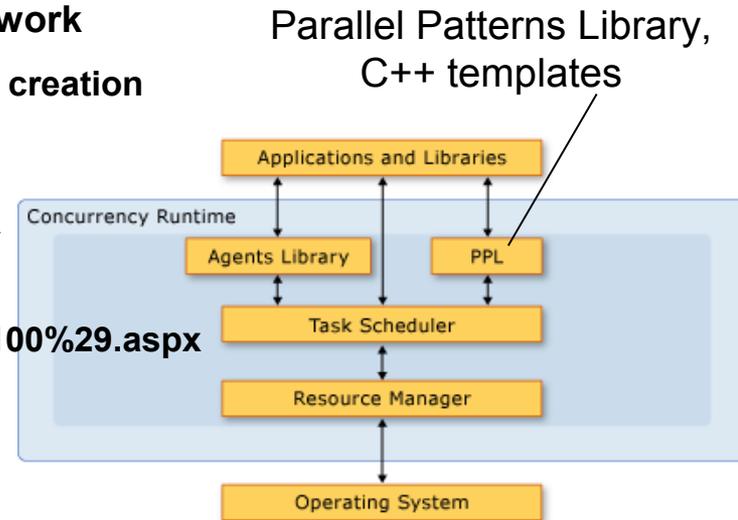
```
// output
OMP Max Threads: 2
OMP Num Cores: 2
Hello from thread 0
Hello from thread 1
```

<http://openmp.org>

<http://en.wikipedia.org/wiki/OpenMP>

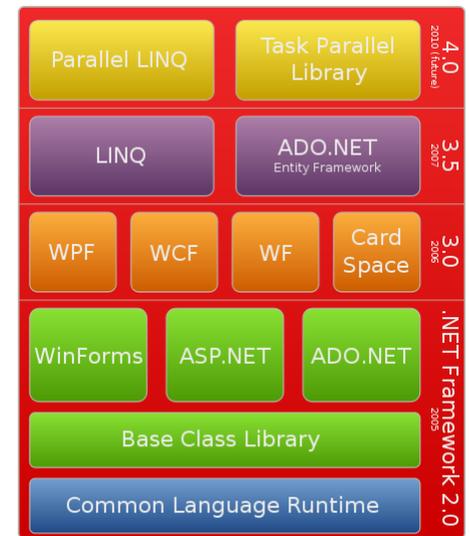
MS VS 2010 and Parallel Computing

- .NET4.0 have Parallel Extensions as a key part of the framework
 - Use of a ThreadPool as in OpenMP eliminate costly thread creation
- Parallel Programming in Native Code (with PPL)
 - <http://blogs.msdn.com/nativeconcurrency/>
- Concurrency Runtime
 - <http://msdn.microsoft.com/en-us/library/dd504870%28VS.100%29.aspx>
- Parallel Programming with .NET
 - <http://blogs.msdn.com/pfxteam/>
- Parallel Computing Developer Center
 - <http://msdn.microsoft.com/en-us/concurrency/default.aspx>



MSDN sites got all info!

```
using System.Threading; // C# - Parallel Ext. work in VB and F# as well
// Equivalent to: for(int i = 0; i < 10; i++)
Parallel.For(0, 10, int i => {
    Console.WriteLine(i);
}); // starting 3 parallel operations
Parallel.Invoke( () => MethodA(), () => MethodB(), () => MethodC() );
// Sequential version of foreach
foreach(var item in sourceCollection) { Process(item); }
// Parallel equivalent of foreach
Parallel.ForEach(sourceCollection, item => Process(item));
```



The .NET Framework Stack

Parallel Java

- Parallel Java has arrived with the Java JDK7 (1.7) release
 - Java has coarse-grained concurrency since JDK5 and threads since start
 - Fork/join framework (fine-grained parallelism)
 - `java.util.concurrent.ForkJoin*` – JSR166y/z (updated revisions)
 - <http://www.javac.info/jsr166z/> - <http://gee.cs.oswego.edu/dl/jsr166/dist/>
- Base class examples
 - RecursiveAction
 - RecursiveTask
 - CyclicAction
 - Parallel***Array
 - AsyncAction

```
/* Creates a ForkJoinPool with a pool size equal to the number
of processors available on the system and using the default
ForkJoinWorkerThreadFactory */
ForkJoinPool fjpool = new ForkJoinPool();
// Creates a new parallel array with given array and executor
ParallelLongArray lpparray =
ParallelLongArray.createUsingHandoff(array, fjpool);
long max = lpparray.max();
```

<http://www.oracle.com/technetwork/java/7-138633.html>

<http://developers.sun.com/learning/javaoneonline/2008/pdf/TS-5515.pdf>

http://www.ddj.com/go-parallel/blog/archives/2009/04/java_7_will_evo.html

ShmooCon 2010

GPU vs. CPU Supercomputing Security Shootout

Collin Brack

You have the fastest Intel/AMD processor in a 500 mile radius thanks to your custom built quad-core, liquid nitrogen cooled, overclocked 5.0Ghz CPU monster. Prepare to be summarily beat down, computationally speaking, by the kid next door who just bought the latest Nvidia GPU to play WOW at 80fps. Video cards, fueled by the gaming industry, have leap-frogged (pun intended) the processing power of the general purpose CPU for certain computational tasks. The rise of the multi-processor based general purpose GPU (GPGPU) platform is taking academia by storm due to its low costs and low barrier to entry into modern day supercomputing. The security community has already embraced the GPU for heavy lifting as have other fields especially when coupled with the sleek marketing efforts by Nvidia and their CUDA development environment, and competing GPU computing platforms from ATI and OpenCL. This 20 minutes session will chronicle the rise of the GPU in high performance computing and will highlight GPU vs. CPU benchmarks of well known security tools including: aircrack (10x speed-up), Pyrit (8x), CUDA Multiforcer, BarsWF MD5 cracker (3x), RainbowCrack multi-GPU CUDA version, and more. Finally, links and tips regarding implementing CUDA in Back|Track 4 are shared.

Collin Brack is a healthcare informatics and medical imaging consultant with experience in computational clusters. He works in academia where he focuses on high performance computing with medical physics researchers. His latest cluster is based on high-end graphics processors to achieve performance gains previously only available to multi-million dollar big iron. He has published and presented on the topics of system design, grid computing, and disaster recovery.

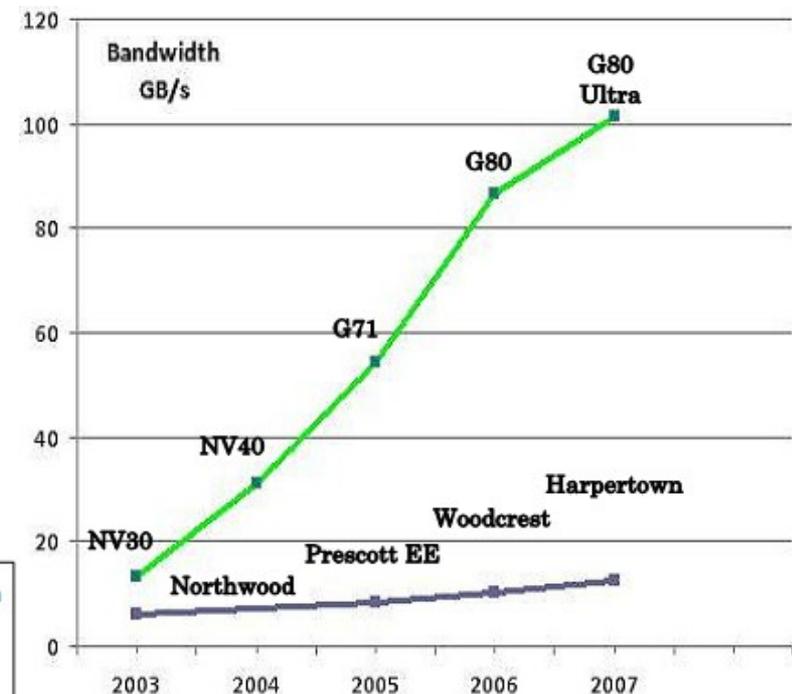
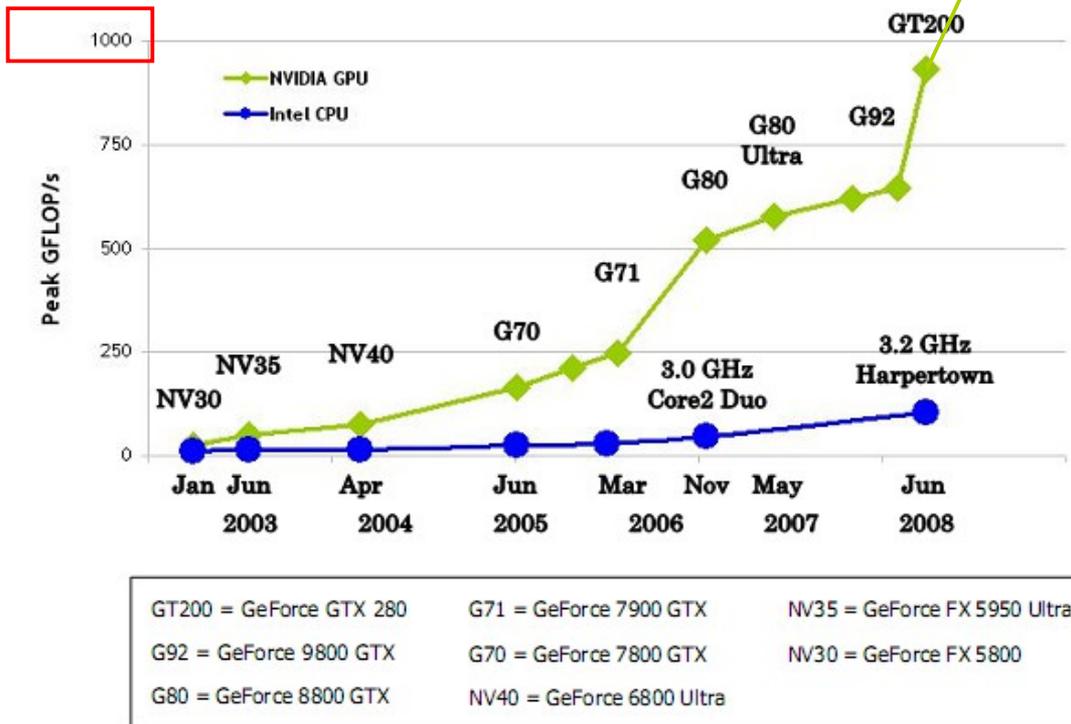
<http://www.shmoocon.org>

GPGPU

We are way off the slide now!

General-Purpose computation on Graphics Processing Units

- The GPU is a **massively parallel** device – **cores / shaders**
 - ATI 5970 got 1600x2 SP (Stream Processors) - 2320x2 GFLOPs single precision
 - A 2.66 GHz Intel Core 2 duo can perform about 25 GFLOPs single precision



NVIDIA Tesla/Fermi - 512x4 SM(Stream Multiprocessors) - 1200x4 GFLOPs single precision

CPU/GPU architecture

- Parallel processor architecture

- Instruction Level Parallelism (ILP) http://en.wikipedia.org/wiki/Instruction_level_parallelism
 - Scalar vs. Superscalar vs. Very Long Instruction Word (VLIW)
 - Instruction pipelining, out-of-order execution, branch prediction, etc.
- Streaming SIMD Extensions (SSE*)

- Flynn's taxonomy

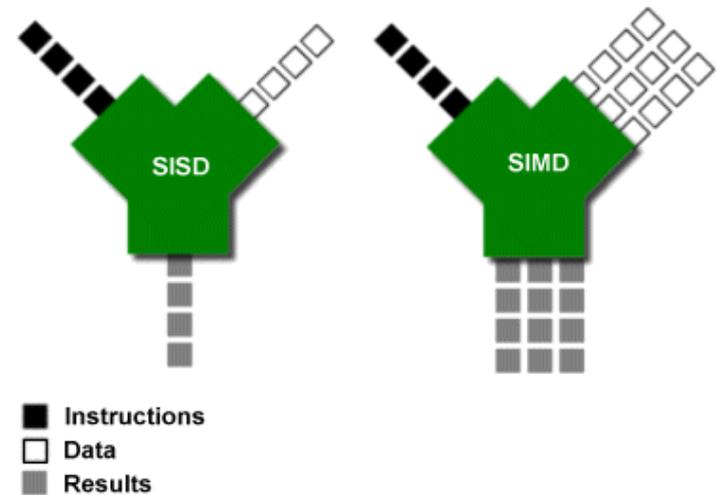
- SISD vs. SIMD vs. MISD vs. MIMD
- http://en.wikipedia.org/wiki/Flynn%27s_taxonomy

- NVIDIA – scalar SIMT (Single Instruction Multiple Thread)

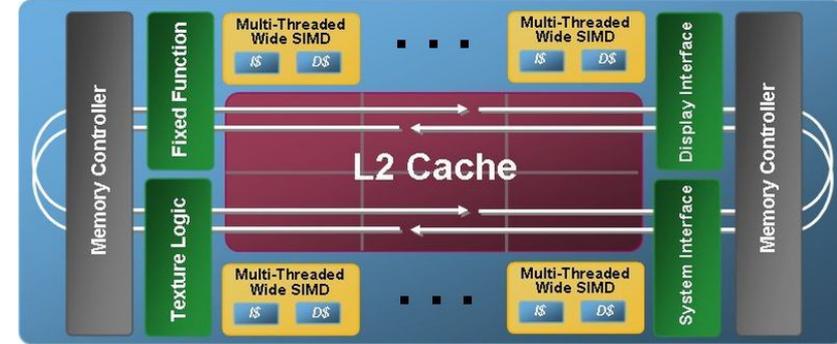
- Brute force approach, simpler compilers, stabler performance

- AMD/ATI - VLIW MIMD (Multiple Instruction Multiple Data)

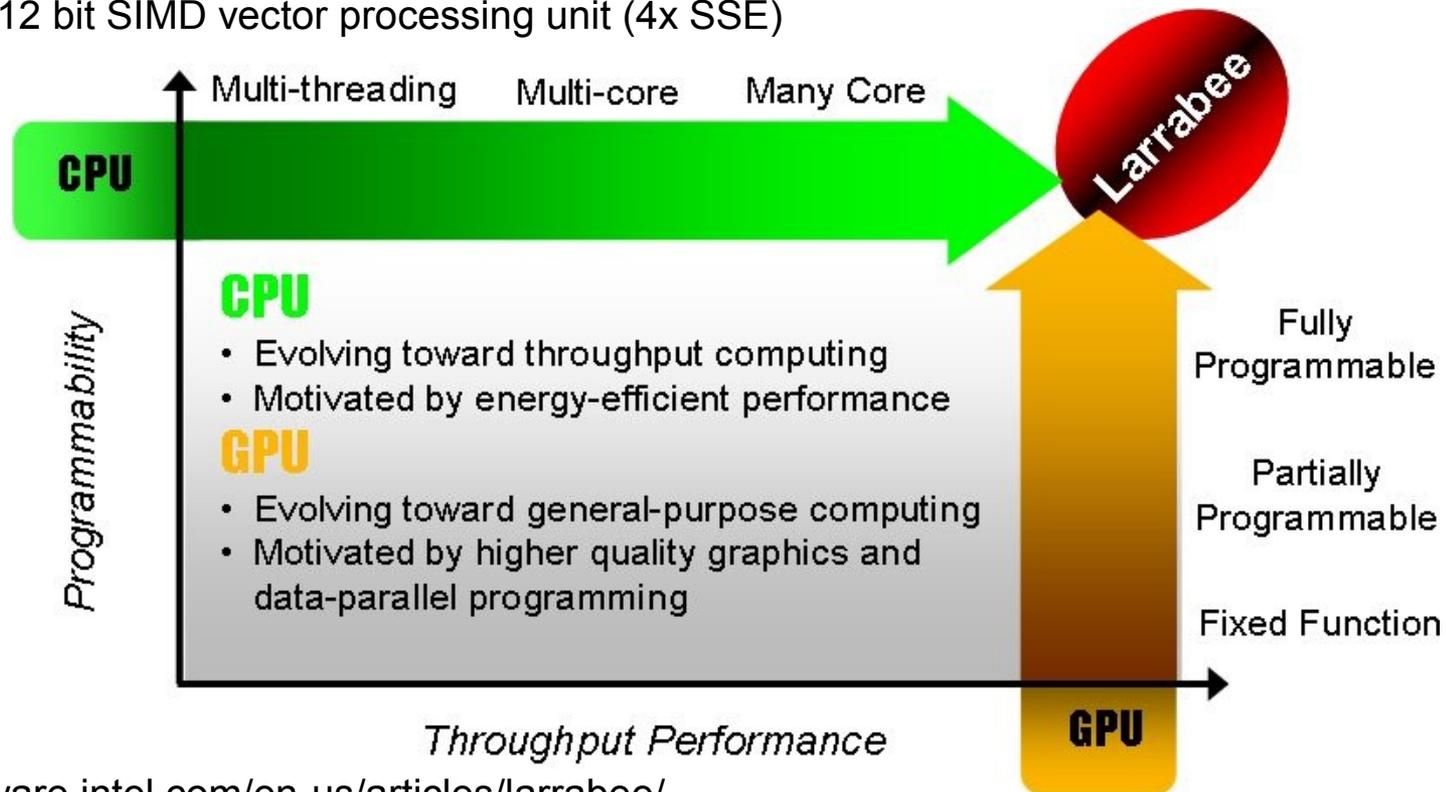
- More advanced/efficient, needs advanced compilers, fall back to SIMD



Intel Larrabee



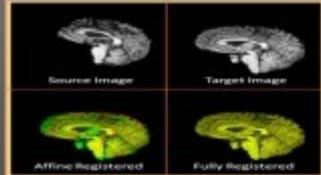
- Differences with current GPUs
 - Very little specialized graphics hardware
 - Use of x86 instruction set, cache coherency across all its cores
- Differences with current CPUs
 - Pentium design with updated features as x64 etc. using at least 32 cores
 - 512 bit SIMD vector processing unit (4x SSE)



NVIDIA CUDA Zone

LATEST CUDA NEWS

Sony Pictures Imageworks Creates Tornado Out Of Spaghetti Sauce With The Help Of NVIDIA Technology



AIRWC

100 x



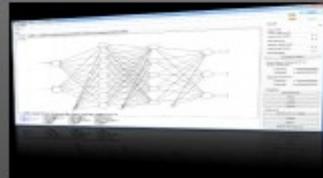
Interactive Visualization of Billion Point Cosmological Simulations

9 x

15 x

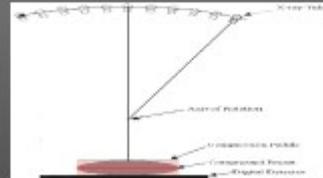


Task and Data Parallel Framework for GPU Computing



Fast pattern classification of ventricular arrhythmias using graphics processing units

53 x



Digital Breast Tomosynthesis Reconstruction

85 x

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Warp schedulers (per SM)	1	1	2
Special Function Units (SFUs) / SM	2	2	4
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache (per SM)	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit



Monte Carlo eXtreme (MCX)

300 x



Scient GPU

the GPU implementation for large scale individual-based simulation of collective

- * Fermi contains 512 CUDA-cores/SM:s (16 SIMT units with 32 cores each).
- * 1 warp = 32 threads which use a minimum of 2 clock cycles.
- * Each SM can handle a maximum of 48 warps and schedule them freely.
- * Track max $512 * 48 = 24576$ threads/GPU!

GPGPU Frameworks - with regard to simplicity

Simplified AMD Stream Computing Programming Model

- Legacy

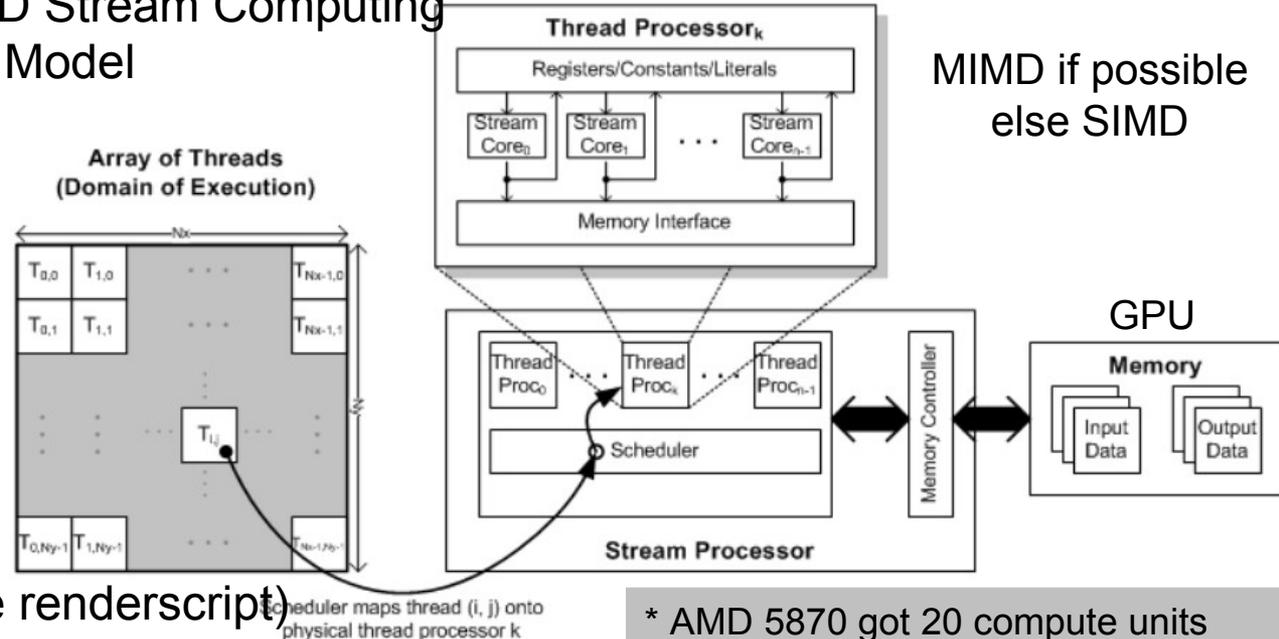
- OpenGL, Direct3D

- Early high level

- Sh/Rapidmind (Intel Ct now)
- BrookGPU (free)
- PeakStream (Google renderscript)
- NVIDIA CUDA (NVCC) - C/C++, Nexus VS plugin
- AMD CAL (CTM, Brook+) - C/C++

- Now

- OpenCL (Open Computing Language) - C (C99) with extensions
 - Platform and hardware independent GP parallel programming
- Microsoft DirectCompute, part of DirectX 11 (10.x will run to)



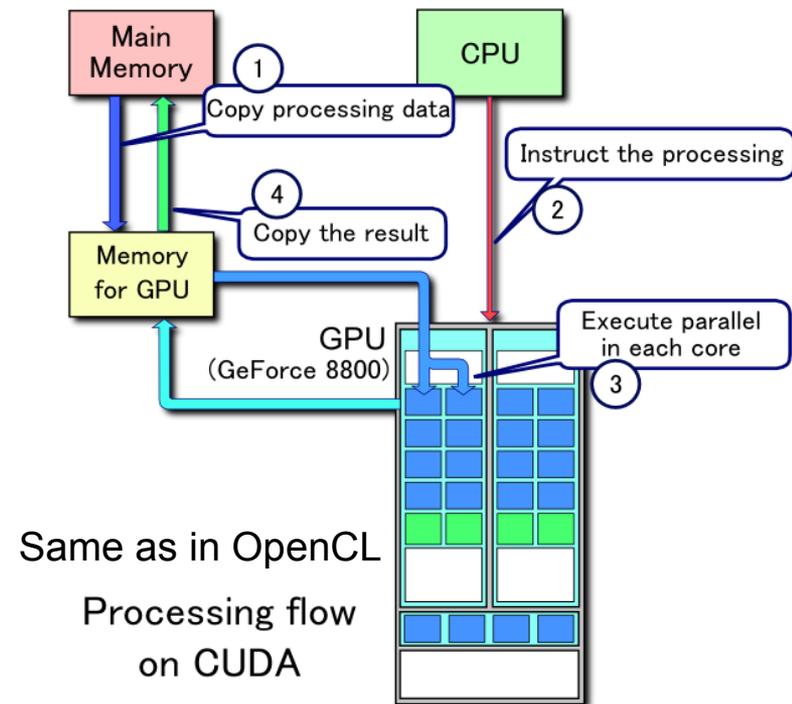
* AMD 5870 got 20 compute units with 16 TPs each, which have a execute unit that can execute 5 different instructions in 1 cycle in a packed VLIW = 1600 SPs
 * 1 wavefront=64 threads @ 4 cycles



OpenCL

OpenCL programming model

- Develop your program with normal source code
- For the parts where you want GPU support you create *_kernel.cl files with OpenCL C-source code
 - The OpenCL code can be embedded in program image as well
- When the program executes the OpenCL driver loads and compile the .cl sources on-demand
 - Binaries can be cached or written to disk avoiding lengthy loads
 - OpenCL scale apps automatically, 1-n CPUs/GPUs etc.
- AMD, nVidia and Intel support OpenCL via the graphics driver
 - Since 2009 - 2011



OpenCL "Hello World" (AMD/Apple)

Runs on: high-performance compute servers, desktop computer systems and handheld devices using a diverse mix of multi-core CPUs, GPUs, Cell-type architectures and other parallel processors such as DSPs.

<https://developer.apple.com/search/?q=opencl>

<http://developer.amd.com/gpu/ATIStreamSDK/pages/TutorialOpenCL.aspx>

```
/* Simple compute kernel which computes the
   square of an input array */
__kernel void square(
    __global float* input,
    __global float* output,
    const unsigned int count)
{
    /* Returns the unique global work-item ID
       value for dimension identified by dimindx */
    int i = get_global_id(0);
    if(i < count)
        output[i] = input[i] * input[i];
}
```

// complement code to the compute kernel

1. Get and select the devices to execute on
 2. Create and open an OpenCL context
 3. Create a command queue to accept the execution and memory requests
 4. Allocate OpenCL memory objects to hold the inputs and outputs for the compute kernel
 5. Online LLVM (Low Level Virtual Machine) compile and build the compute kernel code
 6. Set up the arguments and execution domain
 7. Kick off compute kernel execution
 8. Collect the results
 9. Shutdown and cleanup
- // Note! 6, 7 and 8 may need to iterate if job is big – i.e. have a chance to stop the job...

PyOpenCL

Python bindings/wrappers

Example with kernel right →

Benchmark test with CPU ↘

```
C:\pyopencl-0.91.4\examples>python benchmark-all.py
```

Execution time of test without OpenCL: 23.0160000324 s

Platform name: ATI Stream

Platform profile: FULL_PROFILE

Platform vendor: Advanced Micro Devices, Inc.

Platform version: OpenCL 1.0 ATI-Stream-v2.0.1

Device name: AMD Turion(tm) 64 X2 Mobile Technology TL-60

Device type: CPU

Device memory: 1024 MB

Device max clock speed: 1995 MHz

Device compute units: 2

Execution time of test: 0.00329344 s

Results OK

<http://mathematician.de/software/pyopencl>

```
1 # Hello World in Python with OpenCL
2 import numpy
3 import pyopencl as cl
4
5 hello = "Hello World"
6 a = numpy.empty((len(hello),), dtype=numpy.byte)
7
8 for platform in cl.get_platforms():
9     for device in platform.get_devices():
10         print "Device name: " + device.name
11
12     ctx = cl.Context([device])
13     queue = cl.CommandQueue(ctx)
14     mf = cl.mem_flags
15     dest_buf = cl.Buffer(ctx, mf.WRITE_ONLY, a.nbytes)
16
17     prg = cl.Program(ctx, """
18 #pragma OPENCL EXTENSION cl_khr_byte_addressable_store : enable
19 __constant char hw[] = "Hello World";
20 __kernel void hello(__global char * out)
21 {
22     size_t tid = get_global_id(0);
23     out[tid] = hw[tid];
24 }
25 """)
26
27     try:
28         prg.build()
29     except:
30         print "Error:"
31         print prg.get_build_info(ctx.devices[0],
32             cl.program_build_info.LOG)
33         raise
34
35     prg.hello(queue, a.shape, dest_buf)
36     hw = numpy.empty_like(a)
37     cl.enqueue_read_buffer(queue, dest_buf, hw).wait()
38     world = ""
39     for res in hw:
40         world = world + chr(res)
41     print world
```

AMD CodeXL

<http://developer.amd.com/community/blog/2013/11/08/codexl-1-3-released/>

The screenshot displays the AMD CodeXL BinarySearch interface in Analyze Mode. The main window shows a tree view on the left with the selected kernel 'Kernel - binarySearch' and its analysis results. The right pane displays a table titled 'Analysis generated by emulated execution' with columns for Family, Device, and various performance metrics across different GPU families: Sea Islands (Bonaire, Hawaii), Southern Islands (Capeverde, Pitcairn, Tahiti), and Northern Islands (Cayman, Devastator).

Family	Sea Islands						Southern Islands								
	Bonaire			Hawaii			Capeverde			Pitcairn			Tahiti		
Device	True	False	Both	True	False	Both	True	False	Both	True	False	Both	True	False	Both
ISA branches executed	40	52	52	40	52	52	40	52	52	40	52	52	40	52	52
Cycles	40	52	52	40	52	52	40	52	52	40	52	52	40	52	52
SALUInst	7	9	9	7	9	9	7	9	9	7	9	9	7	9	9
SFetchInst	6	7	7	6	7	7	6	7	7	6	7	7	6	7	7
VALUInsts	7	9	9	7	9	9	7	9	9	7	9	9	7	9	9
VFetchInsts	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2
VWriteInsts	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
LDSInsts	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GDSInsts	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AtomicsInsts	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SGPRs	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
VGPRs	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
WaveFronts	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096

The output window shows the following text:

```
==== Build & Analyze started: Building BinarySearch_Kernels.cl on 7 devices. =====
Compiling device: Bonaire... Succeeded!
Compiling device: Capeverde... Succeeded!
Compiling device: Cayman... Succeeded!
Compiling device: Devastator... Succeeded!
Compiling device: Hawaii... Succeeded!
Compiling device: Pitcairn... Succeeded!
Compiling device: Tahiti... Succeeded!
```

- GPU Debugger: Debugging tool for OpenCL™/OpenGL™ API calls and OpenCL™ kernels
- CPU Profiler: A profiling suite for tuning application performance for AMD CPUs
- GPU Profiler: A GPU profiler for OpenCL™ and DirectCompute applications on AMD APUs/GPUs
- Static Analyzer: Analyze OpenCL™ kernels statically to estimate and tune performance

GPGPU testing 1

- IGHASHGPU (Brook+/CUDA), recover/crack SHA1, MD5 & MD4 hashes
 - Supports salted hashes, NTLM, MySQL*, Oracle 11g, ..., etc.
 - Plain MD5, 8 chars, lowercase
 - Windows 7 x64, AMD Phenom x4 @ 2.2GHz
 - ATI 4850 - 800SP, Catalyst 9.12, Stream SDK v2.0
 - Count down time (ETA) started at almost 4 minutes

<http://golubev.com>
Intresting discussion
RAR GPU as well

```
Administrator: C:\Windows\system32\cmd.exe - test.cmd
C:\Users\hjo\Desktop\OpenCL\ighashgpu_v062>ighashgpu.exe /h:E8DC4081B13434B45189A720B77B6818 /t:md5 /c:s /max:8
*****
***          MD4/MD5/SHA1 GPU Password Recovery v0.62          ***
***  For ATI RV 7X0 cards and nVidia 'CUDA' ones (G80+)      ***
***  (c) 2009 Ivan Golubev, http://golubev.com                ***
***  see "readme.htm" for more details                        ***
*****
*** Any commercial use of this program is strictly forbidden ***
*****
Found 1 CAL device(s)
Starting brute-force attack, Charset Len = 26, Min passlen = 4, Max passlen = 8
Charset (unicode -> 0) [abcdefghijklmnopqrstuvwxyz]
Charset in HEX: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a
Starting from [aaaa]
Hash type: MD5, Hash: e8dc4081b13434b45189a720b77b6818
Device #0: [RV7x0] 625.00 Mhz 800 SP
Hardware monitoring enabled, threshold temperature is 90°C.
CURPWD: ygrlegns DONE: 16.35% ETA: 3m 1s AURSPD: 960.9M
Found password: [abcdefgh], HEX: 61 62 63 64 65 66 67 68
Processed 42 505 076 736 passwords in 45s.
Thus, 961 066 243 password(s) per second in average.
```

BF NT hash crack
7 char pass a-z,0-9
ETA:
Cain, 3.5h
IGHASHGPU, 1 min

**Current GPU
generation is
more than 10
times faster!**

GPGPU testing 2

- BarsWF (Brook+/CUDA/SSE), recover/crack MD5
 - Same settings as for IGHASHGPU

DirectCompute
Benchmark v0.44b

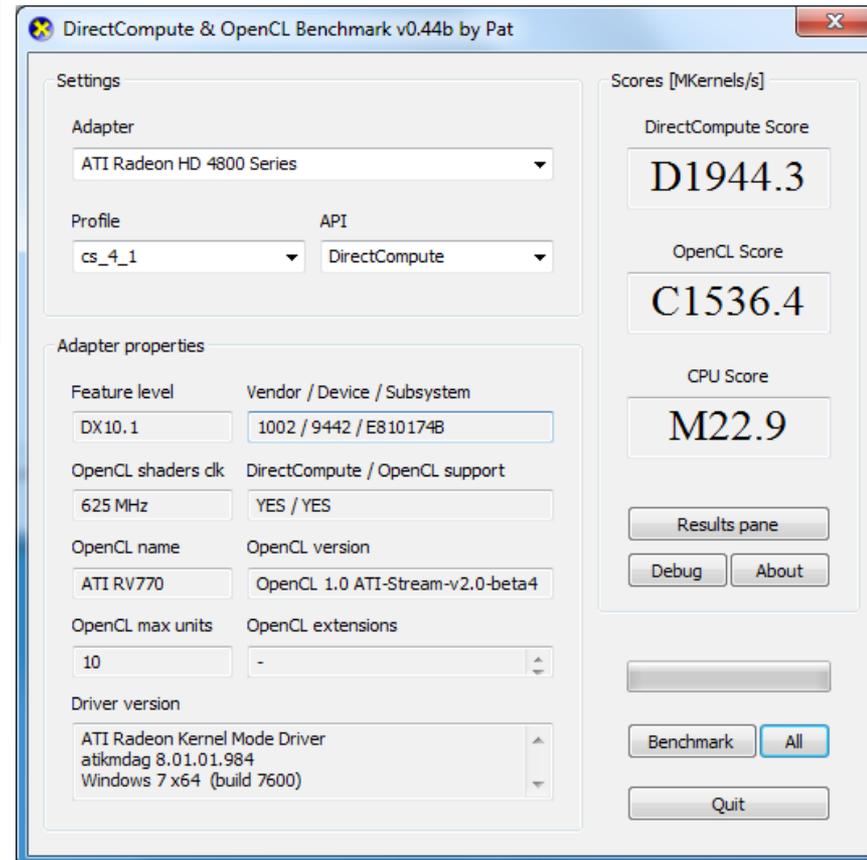
```
Administrator: C:\Windows\system32\cmd.exe - BarsWF_Brook_x64.exe -h E8DC4081B13
BarsWF MD5 bruteforcer v0.9b♥ http://3.14.by/en/md5
by Svarychevski Michail http://3.14.by/ru/md5

GPU0 : 866.05 MHash/sec CPU0 : 22.00 MHash/sec
CPU1 : 21.78 MHash/sec
CPU2 : 21.59 MHash/sec

GPU* : 866.05 MHash/sec CPU* : 65.37 MHash/sec

Key: eycdgpae Avg.Total: 931.42 MHash/sec
Hash: E8DC4081B13434B45189A720B77B6818
Progress: 8.17 % ETC 0 days 0 hours 3 min 25 sec

Key is: abcdefgh
Press any key to exit_
```



- Others
- OCLCrack (OpenCL) with source
- <http://sghctoma.extra.hu/index.php?p=entry&id=11>
- Multihash Bruteforcer CUDA
- <http://www.cryptohaze.com/>
- Extreme GPU Bruteforcer CUDA
- <http://www.insidepro.com/eng/egb.shtml>

MD5 Software Benchmark

- From the Code Breaker blog

Marc Bevand have some interesting articles about breaking crypto using ATI CAL IL and Sony PS3

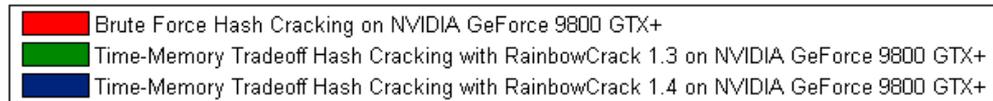
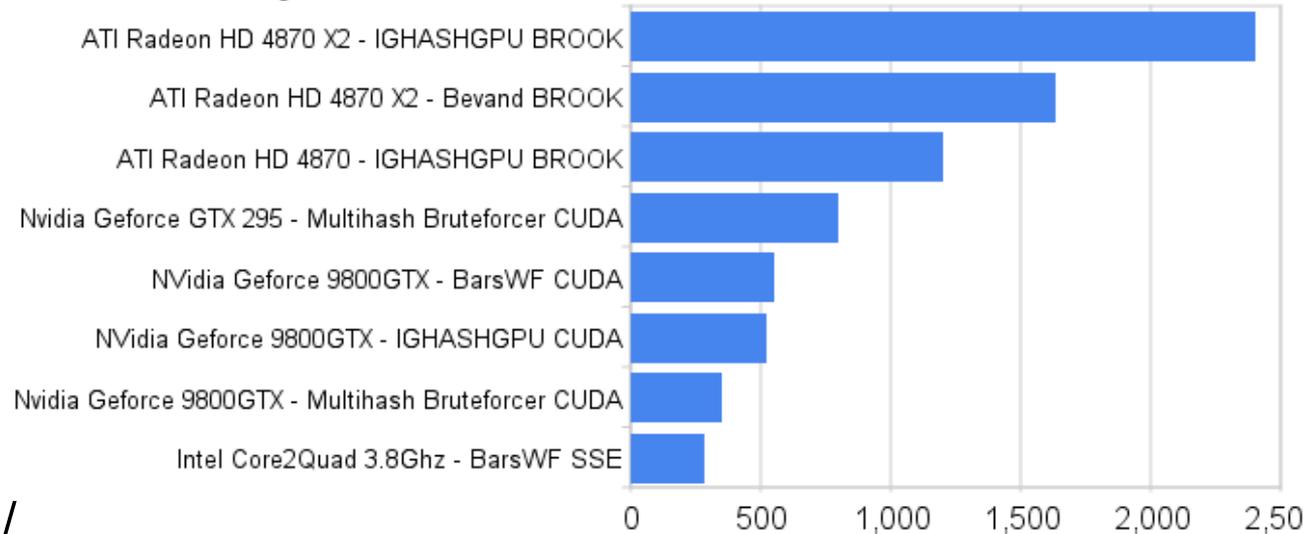
<http://www.zorinaq.com/>

RainbowCrack performance

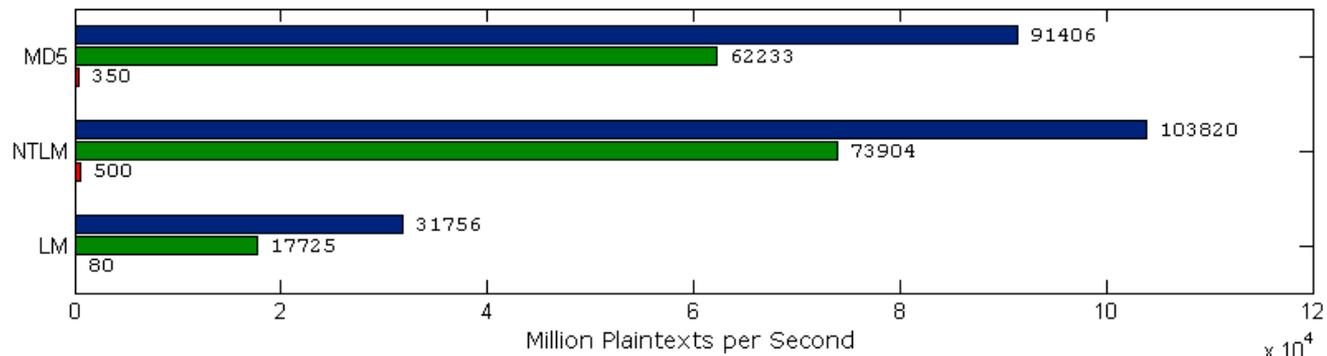
Around 200 times faster than brute force with GPU

Run tables with CPU?

MD5 Software SPEED million plaintexts/sec



Hash Cracking Performance



GPU limitations

- System to GPU bandwidth is limited
 - Around 1 byte per clock tick
 - A CPU will be able to perform **A LOT** of instructions during the copy of input and output data to/from the compute kernel
- Error control handling and debugging can be difficult to perform
- GPUs like their data arranged in specific ways/formats etc.
- Not all problems/algorithms are optimal to run on a GPU
- Amdahls law
 - How much in a program can be parallelized?

http://en.wikipedia.org/wiki/Amdahl%27s_law

$$\text{speedup} = \frac{1}{\frac{P}{N} + S}$$

where P = parallel fraction, N = number of processors and S = serial fraction

N	speedup		
	P = .50	P = .90	P = .99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02
100000	1.99	9.99	99.90

GP GPU references/resources

- Blogs

- Speed Junkie - <http://gpgpu-computing.blogspot.com/>
- Code Breaker - <http://jchblue.blogspot.com/>
- GPU computing - <http://oscarbg.blogspot.com/>

- OpenCL tutorials etc.

- <http://www.macresearch.org/openc1>
- <http://gpgpu.org/>
- <http://en.wikipedia.org/wiki/DirectCompute>
- <http://developer.amd.com>
- <http://developer.nvidia.com>
- <http://www.khronos.org/developers/>
- <http://www.geeks3d.com/>

- C# bindings/wrappers - <http://sourceforge.net/projects/cloo/>

- Java bindings/wrappers - <http://www.jocl.org/>



Field-programmable gate array (FPGA)

- Tableau TACC1441 Hardware Accelerator \$3900
 - <http://www.tableau.com/>
 - AccessData PRTK support - TACC_Install.pdf
 - <http://www.digitalintelligence.com/products/rack-a-tacc/>
- Bruce Schneier - Secure Passwords Keep You Safer
 - <http://www.schneier.com/essay-148.html>
- NSA (at) Home
 - Breaks 800 hashes concurrently
 - <http://nsa.unaligned.org/>



Rack-A-TACK 2U module, \$20000

Rack-A-TACC™

Performance Data

