4 Sessions

Marian HackMan Marinov

OpenFest

1st

Increasing the performance using SSE, AVX* and FMA extensions

- AVX - Advanced Vector Extensions
- AVX2 - 256bit integers
  - FMA - Fused multiply-accumulate
- AVX-512 - 512bit integers
- SSE - Streaming SIMD Extensions

  SIMD - Single Instruction Multiple Data

- Exploit AVX for matrix multiplication
- Exploit SSE

  - for binary operations on multiple inputs

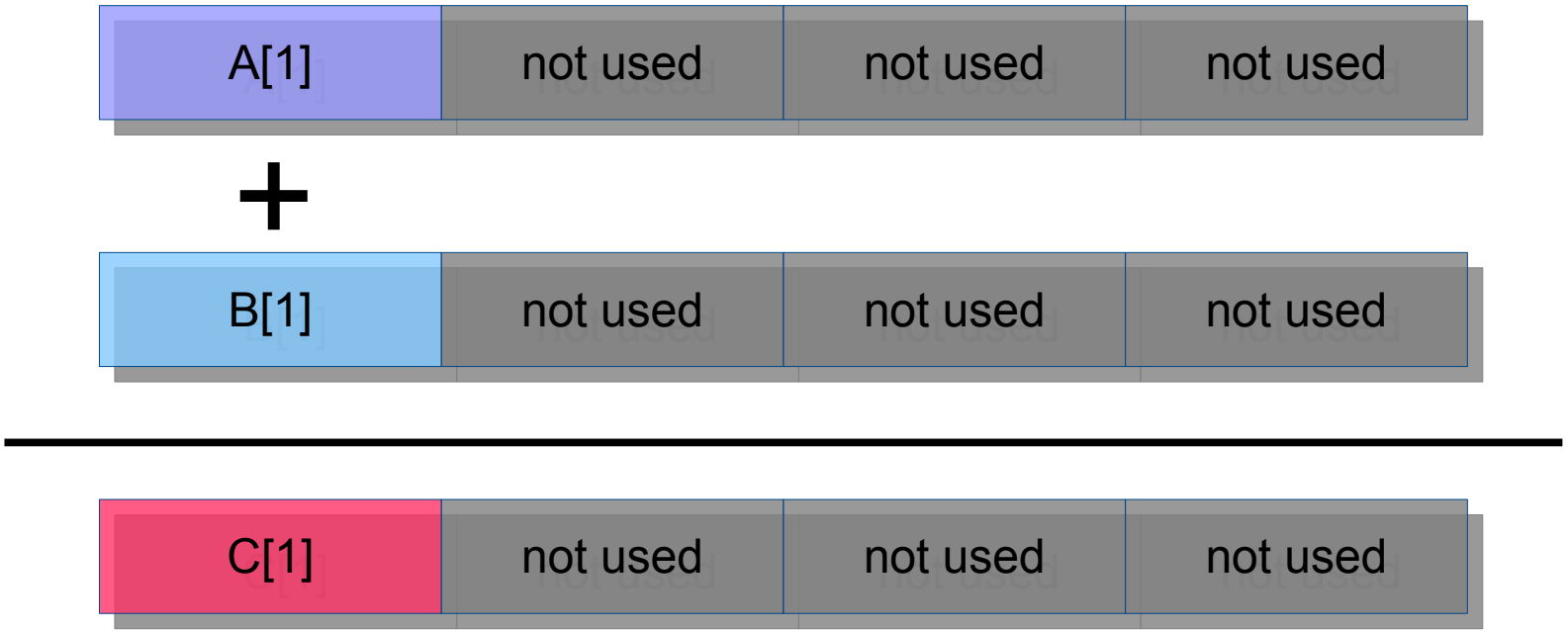
  - for populating multiple registers with single instructions
- AVX-512 for prefetching data

Why does it work?

- Vectorization

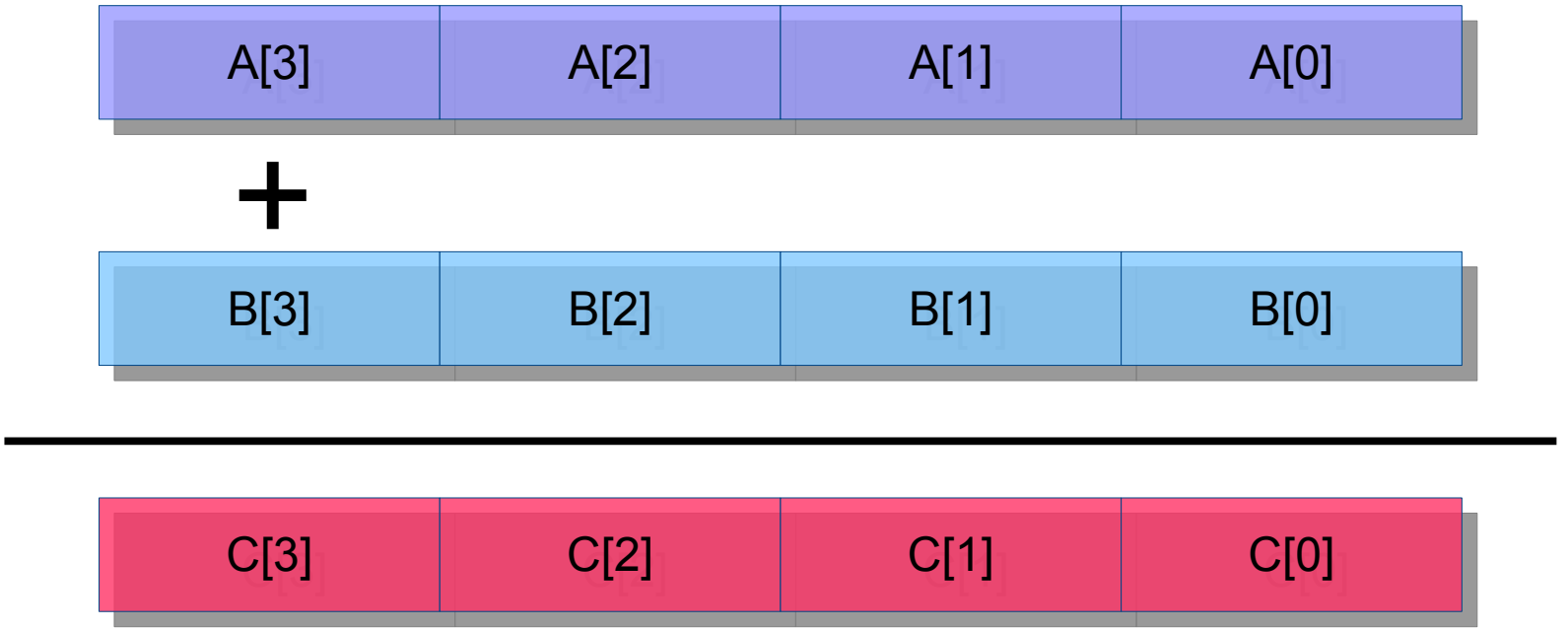
```
#define MAX 1000000
int a[256], b[256], c[256];
int main () {
  int i,j;
  for (j=0; j<MAX; j++){
    for (i=0; i<256; i++){
      a[i] = b[i] + c[i];
    }
    return 0;
  }
```

Why does it work?



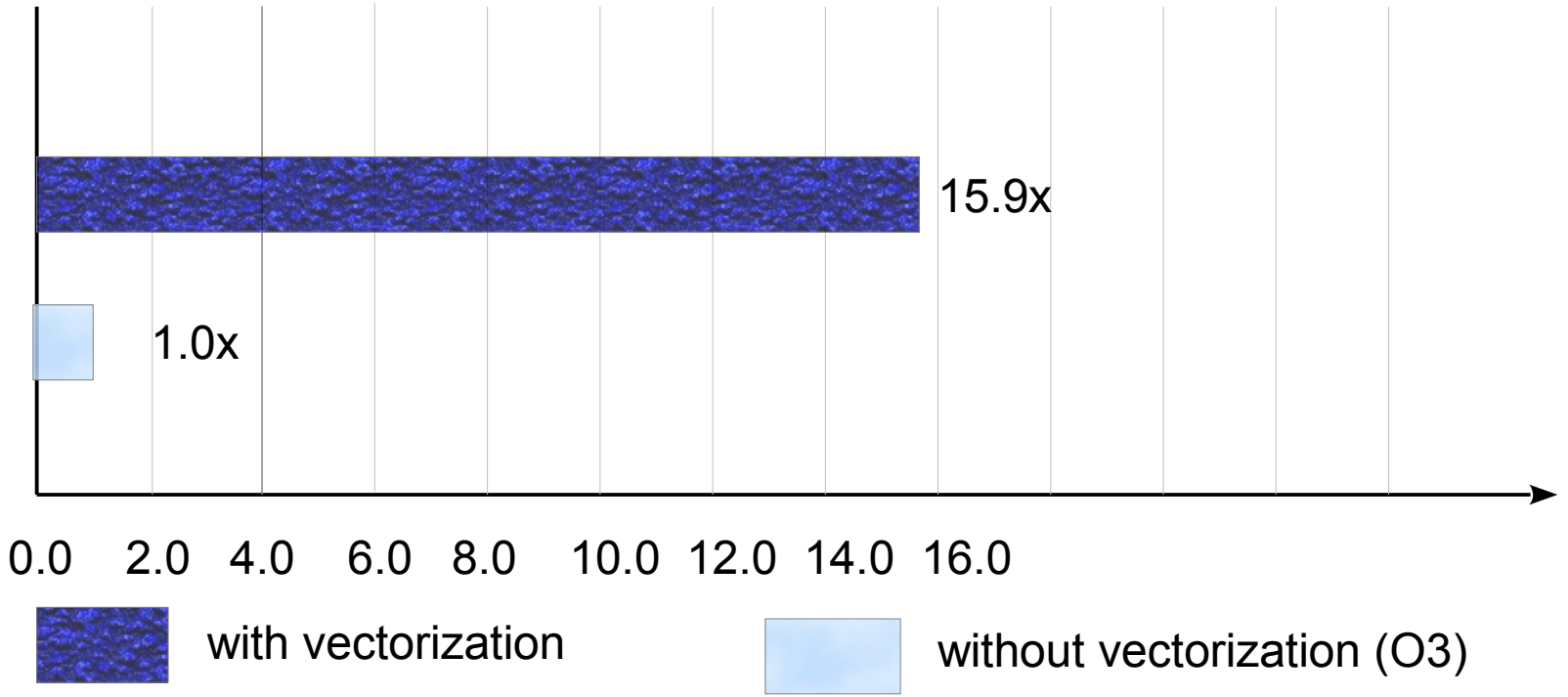
3x 32-bit unused integers

Why does it work?



Why does it work?

```
$ gcc -fopt-info-vec sort.c –O2 –ftree-vectorize
$ gcc -fopt-info-vec sort.c –O3
```



https://github.com/VictorRodriguez/autofdo_tutorial/blob/master/sort.c

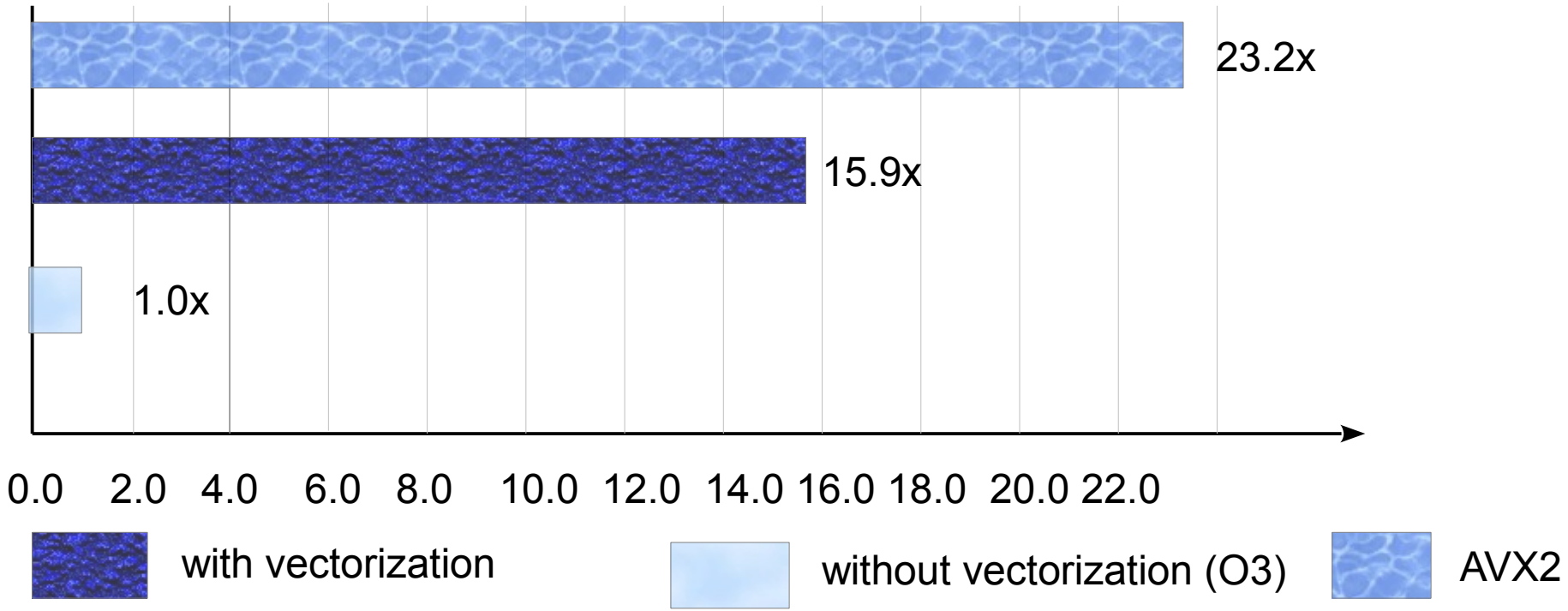
AVX-512

```
$ gcc -O3 sanity.c -fopt-info-vec -mavx2 -o sanity
```

511 -> 256	255 -> 128	127 -> 0
Intel AVX 512	Intel AVX2/ Intel AVX	SSE
XMM0	YMM0	ZMM0
XMM1	YMM1	ZMM1
XMM2	YMM2	ZMM2
XMM3	YMM3	ZMM3
XMM4	YMM4	ZMM4
XMM5	YMM5	ZMM5
XMM6	YMM6	ZMM6

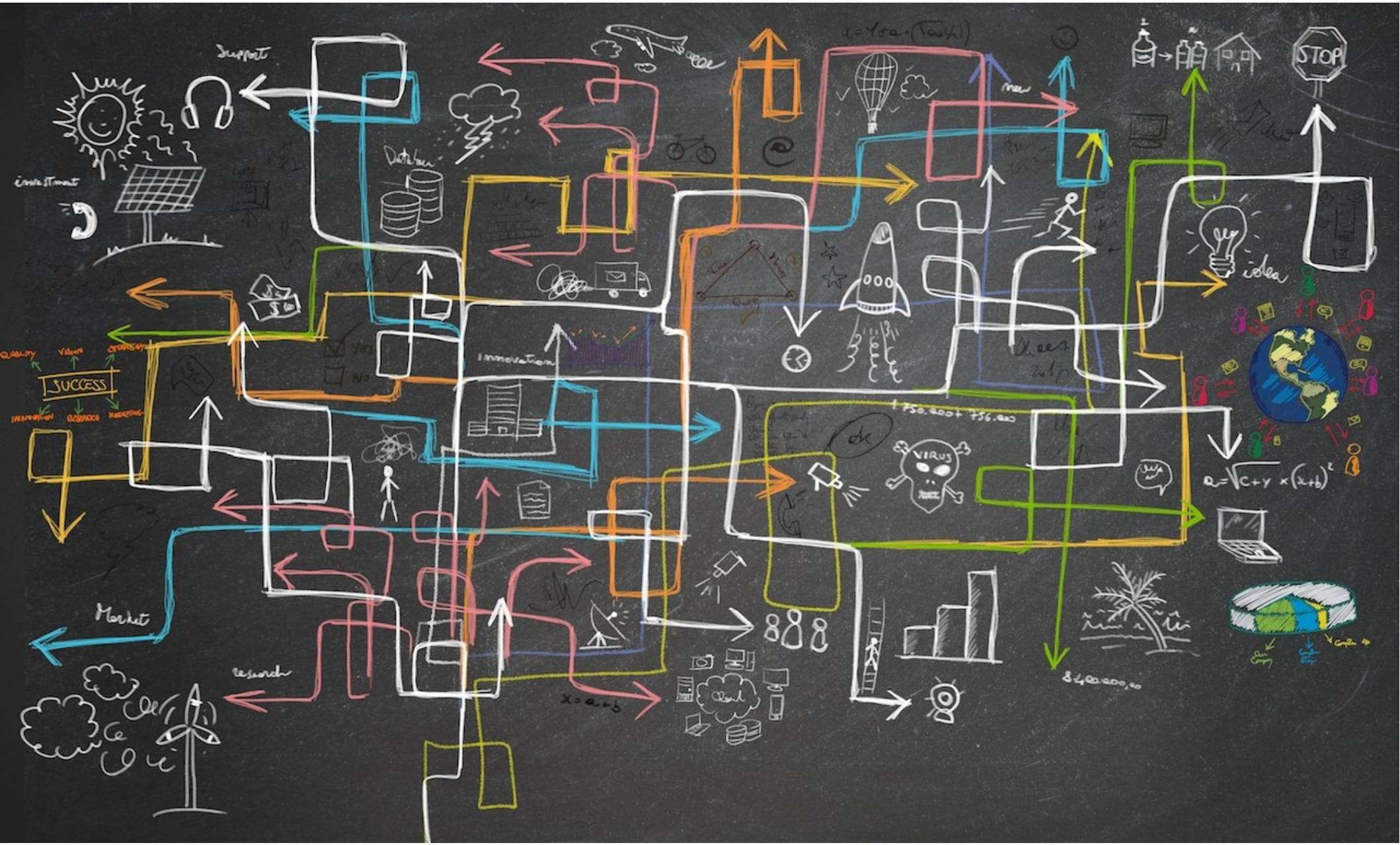
Why does it work?

```
$ gcc -O3 sanity.c -fopt-info-vec -mavx2 -o sanity
```



https://jobs.siteground.bg/#devops

It's complicated



https://jobs.siteground.bg/#devops

Intel Clear Linux

* Modified glibc
* Modified Python package
* Modified R package

https://github.com/clearlinux/make-fmv-patch
https://github.com/clearlinux-pkgs
https://clearlinux.org/

https://jobs.siteground.bg/#devops

2nd

BPF BCC tools
for performance analysis

What is BPF?

```
# tcpdump host 127.0.0.1 and port 22 -d
(000) ldh      [12]
(001) jeq      #0x800           jt 2    jf 18
(002) ld       [26]
(003) jeq      #0x7f000001      jt 6    jf 4
(004) ld       [30]
(005) jeq      #0x7f000001      jt 6    jf 18
(006) ldb      [23]
(007) jeq      #0x84            jt 10   jf 8
(008) jeq      #0x6             jt 10   jf 9
(009) jeq      #0x11            jt 10   jf 18
(010) ldh      [20]
(011) jset     #0x1fff          jt 18   jf 12
(012) ldxb     4*([14]&0xf)
(013) ldh      [x + 14][...]
```

Optimizes packet filter performance

2 x 32-bit registers & scratch memory

User-defined bytecode executed by an in-kernel sandboxed virtual machine

Steven McCanne and Van Jacobson, 1993

What is eBPF?

```
/* Register numbers */
enum {
    BPF_REG_0 = 0,
    BPF_REG_1,
    BPF_REG_2,
    BPF_REG_3,
    BPF_REG_4,
    BPF_REG_5,
    BPF_REG_6,
    BPF_REG_7,
    BPF_REG_8,
    BPF_REG_9,
    BPF_REG_10,
    __MAX_BPF_REG,
};
```

10 x 64-bit registers

maps (hashes)

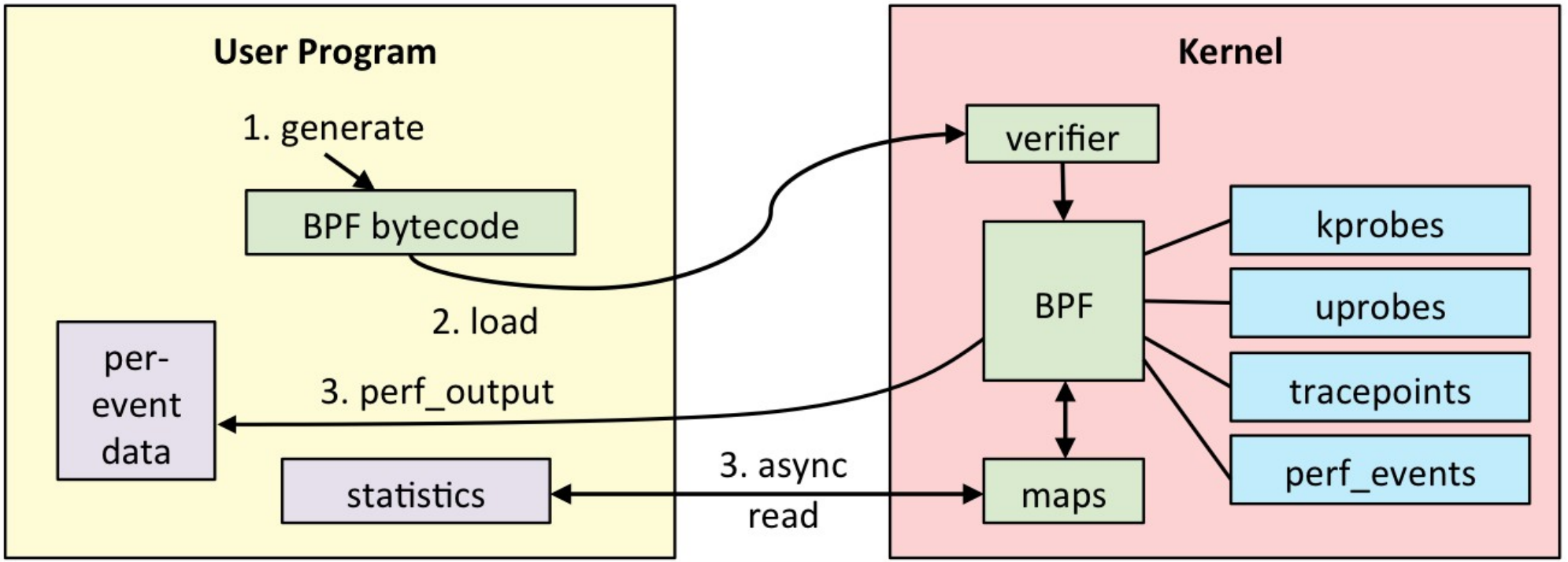
actions

What is eBPF?

```
struct bpf_insn prog[] = {
  BPF_MOV64(BPF_REG_6, BPF_REG_1),
  BPF_LD_ABS(BPF_B, ETH_HLEN + offsetof(struct iphdr, protocol), /* R0 = ip->proto */
  BPF_STX_MEM(BPF_W, BPF_REG_10, BPF_REG0, -4), /* *(u32 *)(fp - 4) = R0 */
  BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),
  BPF_ALU64_IMM(BPF_ADD, BPF_REG_2, -4),       /* R2 = fp - 4 */
  BPF_LD_MAP_FDD(BPF_REG_1, map_fd),
  BPF_RAW_INSN(BPF_JMP | BPF_CALL, 0, 0,
BPF_FUNC_map_lookup_elem),
  BPF_JMP_IMM(BPF_JEQ, BPF_REG_0, 0, 2),
  BPF_MOV64_IMM(BPF_REG_1, 1),                 /* R1 = 1 */
  BPF_RAW_INSN(BPF_STX | BPF_XADD | BPF_DW, BPF_REG_0,
BPF_REG_1, 0, 0),                              /* xadd R0 += R1 */
  BPF_MOV64_IMM(BPF_REG_0, 0),                 /* R0 = 0 */
  BPF_EXIT_INSN(),
};
```

https://jobs.siteground.bg/#devops

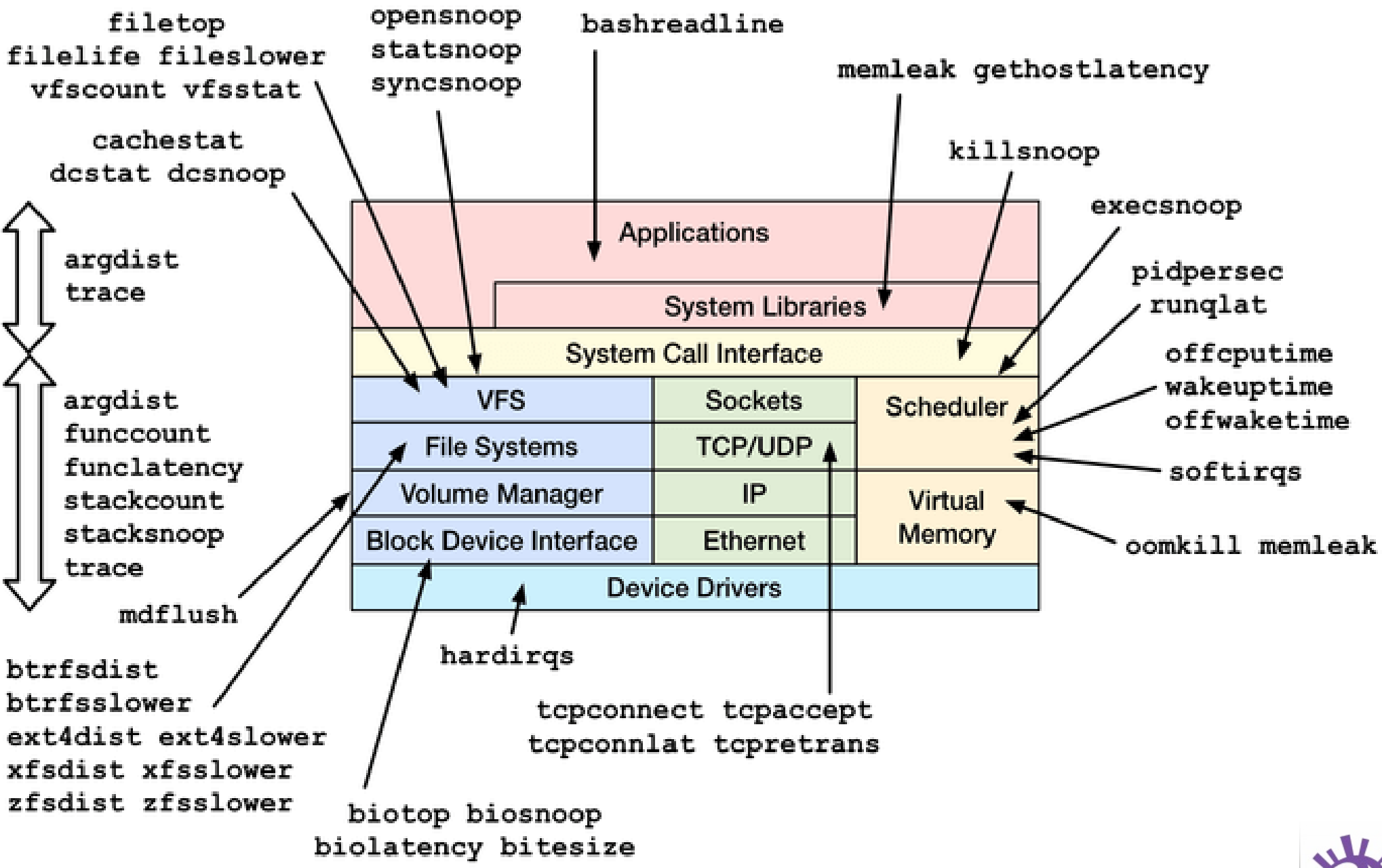
How does it work?



https://jobs.siteground.bg/#devops

What else can you do with it?

Linux bcc/BPF Tracing Tools



https://jobs.siteground.bg/#devops

https://github.com/iovisor/bcc#too

Where are these tools?

https://github.com/iovisor/bcc

Brendan Gregg

Senior Performance Architect, Netflix

Some examples

```
# ./execsnoop
PCOMM            PID    RET ARGS
supervise        9660     0 ./run
supervise        9661     0 ./run
mkdir            9662     0 /bin/mkdir -p ./main
run              9663     0 ./run
[...]
```


Some examples

```
# ./execsnoop
PCOMM            PID    RET ARGS
supervise        9660     0 ./run
supervise        9661     0 ./run
mkdir            9662     0 /bin/mkdir -p ./main
run              9663     0 ./run
[...]
```


Some examples

```
# ./opensnoop
PID    COMM       FD ERR PATH
1565   redis-server       5   0 /proc/1565/stat
1565   redis-server       5   0 /proc/1565/stat
1565   redis-server       5   0 /proc/1565/stat
1603   snmpd              9   0 /proc/net/dev
1603   snmpd             11   0 /proc/net/if_inet6
1603   snmpd             -1   2 /sys/class/net/eth0/device/vendor
1603   snmpd             11   0
/proc/sys/net/ipv4/neigh/eth0/retrans_time_ms
1603   snmpd             11   0
/proc/sys/net/ipv6/neigh/eth0/retrans_time_ms
1603   snmpd             11   0
/proc/sys/net/ipv6/conf/eth0/forwarding
[...]
```

Some examples

```
# ./cachestat
```

HITS	MISSES	DIRTIES	READ HIT%	WRITE HIT%	BUFFERS MB	CACHED
1074	44	13	94.9%	2.9%	1	223
2195	170	8	92.5%	6.8%	1	143
182	53	56	53.6%	1.3%	1	143
62480	40960	20480	40.6%	19.8%	1	223
7	2	5	22.2%	22.2%	1	223
348	0	0	100.0%	0.0%	1	223

[...]

Some examples

```
# ./biolatency
Tracing block device I/O... Hit Ctrl-C to end.
^C
     usecs               : count     distribution
       0 -> 1            : 0        |                                        |
       2 -> 3            : 0        |                                        |
       4 -> 7            : 0        |                                        |
       8 -> 15           : 0        |                                        |
      16 -> 31           : 0        |                                        |
      32 -> 63           : 0        |                                        |
      64 -> 127          : 1        |                                        |
     128 -> 255          : 12       |********                                |
     256 -> 511          : 15       |**********                              |
     512 -> 1023         : 43       |*****************************           |
    1024 -> 2047         : 52       |************************************    |
    2048 -> 4095         : 47       |********************************        |
    4096 -> 8191         : 52       |************************************    |
    8192 -> 16383        : 36       |************************                |
   16384 -> 32767        : 15       |**********                              |
   32768 -> 65535        : 2        |*                                       |
   65536 -> 131071       : 2        |*                                       |
```

Some examples

```
# ./biosnoop
TIME(s)      COMM       PID  DISK  T SECTOR    BYTES  LAT(ms)
0.000004001 supervise  1950 xvda1 W 13092560  4096   0.74
0.000178002 supervise  1950 xvda1 W 13092432  4096   0.61
0.001469001 supervise  1956 xvda1 W 13092440  4096   1.24
0.001588002 supervise  1956 xvda1 W 13115128  4096   1.09
1.022346001 supervise  1950 xvda1 W 13115272  4096   0.98
1.022568002 supervise  1950 xvda1 W 13188496  4096   0.93
[...]
```


Some examples

```
# ./runqlat
Tracing run queue latency... Hit Ctrl-C to end.
     usecs               : count     distribution
         0 -> 1          : 233      |***********                             |
         2 -> 3          : 742      |************************************    |
         4 -> 7          : 203      |**********                              |
         8 -> 15         : 173      |********                                |
        16 -> 31         : 24       |*                                       |
        32 -> 63         : 0        |                                        |
        64 -> 127        : 30       |*                                       |
       128 -> 255        : 6        |                                        |
       256 -> 511        : 3        |                                        |
       512 -> 1023       : 5        |                                        |
      1024 -> 2047       : 27       |*                                       |
      2048 -> 4095       : 30       |*                                       |
      4096 -> 8191       : 20       |                                        |
      8192 -> 16383      : 29       |*                                       |
     16384 -> 32767      : 809      |****************************************|
     32768 -> 65535      : 64       |***
```

3rd

Insecurity of today's computers.

Ring -2 firmware and UEFI, and why we wouldn't want them

https://jobs.siteground.bg/#devops

4th

Comparison between the functionality of the best known Nginx distributions

Nginx, *OpenResty* and Tengine

https://jobs.siteground.bg/#devops

Nginx is one of the fastest web servers in the world

https://jobs.siteground.bg/#devops

How to get it?

- Distribution package
- other repos with prebuild packages

How to get it?

- Manual compilation
- go with Nginx plus

Alternatives?

- OpenResty
- Tengine

OpenResty

- OpenResty® is a dynamic web platform based on NGINX and LuaJIT.
- a good source for high quality Nginx modules
- 25 different nginx modules

https://openresty.org/en/

https://github.com/openresty/

OpenResty

* highlights:
  - sregex
  - headers-more
    - clear headers on input
    - clear or replace headers on output
  - replace-filter
    - regexp replace BODY filter

https://jobs.siteground.bg/#devops

OpenResty

I believe that it is the best
web application platform
you can directly use

https://jobs.siteground.bg/#devops

Tengine

- this is the web server that Alibaba runs on
- its main purpose is performance
- its a collection of different nginx modules

Tengine

* Proxy/Load balancing
  - Dynamic Upstream updates
  - Upstream domain resolver
  - Limit upstream tries
  - Upstream check module
  - Upstream keepalive timeout
  - Consistent hash module
  - Session sticky module
  - Slice module

Tengine

* Filters
  - Concat
  - Headers
  - Footer
  - Trim
  - Reqstat
  - TFS
  - User agent

Conclusion

- if you are preparing a load balancer/proxy, go with Tengine

- it you are preparing a web application server, go with OpenResty

https://jobs.siteground.bg/#devops

Thank you!

https://jobs.siteground.bg/#devops

SiteGround

Marian HackMan Marinov
mm@siteground.com