JOHN-THE-RIPPER - MAKING IT BETTER

John the Ripper (JTR) is a brilliant offline password cracking tool (found here) and if you get the jumbo version with the community patches, it cracks a large variety of hashes and quite nicely. But can it be done better? Lets start with a plain and simple build:

```
#make generic
```

..and then lets run a test. We will use DES for our purposes:

```
# ./john -format=des --test
Benchmarking: Traditional DES [32/32 BS]... DONE
Many salts:    323614 c/s real, 321629 c/s virtual
Only one salt:    336061 c/s real, 336061 c/s virtual
```

What this shows us is the number of calculations per second that this build of john can process. Looks quite impressive on my little dual core setup, but as mentioned can we make it better? If you have a look at the options avaliable to the JTR *Makefile*, you can match those options to the capabilities of your CPU. So lets see what it does for us:

```
#make linux-x86-sse2
# ./john -format=DES --test
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts:    1700K c/s real, 1700K c/s virtual
Only one salt:    1650K c/s real, 1634K c/s virtual
```

Yes, yes it does make a difference. And quite a difference. But can it be better? You see by default JTR only uses one CPU or CPU core. Not all possible CPU power. We can change that, you see the latest versions of JTR can be setup to use a MPI (distributed computing) setup. MPI is normally used in clusters across multiple computer nodes, but it can also be used to increase the load across multiple cores. It takes a bit more work:

```
#get mpich2 and mod (apt-get / yum / etc)
#get /etc/mpd.conf or user based con (running mod will give you instructions to do
this)
#start mpd with mpd -d
```

..then in the *john/src/Makefile*, make this:

```
## Uncomment the TWO lines below for MPI (can be used together with OMP as well)
## If you experience problems with MPI_Barrier, remove -DJOHN_MPI_BARRIER
## If you experience problems with MPI_Abort, remove -DJOHN_MPI_ABORT
#CC = mpicc -DHAVE_MPI -DJOHN_MPI_BARRIER -DJOHN_MPI_ABORT
#MPIOBJ = john-mpi.o
```

..to look like this:

```
## Uncomment the TWO lines below for MPI (can be used together with OMP as well)
## If you experience problems with MPI_Barrier, remove -DJOHN_MPI_BARRIER
## If you experience problems with MPI_Abort, remove -DJOHN_MPI_ABORT
CC = mpicc -DHAVE_MPI -DJOHN_MPI_BARRIER -DJOHN_MPI_ABORT
MPIOBJ = john-mpi.o
```

Now run the tailored make again:

```
#make linux-x86-sse2
```

Then use *mpiexec -np <number of cores>*, I have a dual core setup so,

```
# mpiexec -np 2 ./john -format=DES --test
Benchmarking: Traditional DES [128/128 BS SSE2]... (2xMPI) DONE
Many salts:    3752K c/s real, 3869K c/s virtual
Only one salt:    3100K c/s real, 3229K c/s virtual
```

It seems to be even better. But does it actually work? Lets compare using a standard set of parameters; (1) a password dictionary of 1 million and (2) a list of 10 DES passwords. First the non-MPI version:

```
# time ./john --rules --wordlist=/admin/002_dict/001_general/test.dic /admin/001_to-
crack/demo.txt
Loaded 10 password hashes with 10 different salts (Traditional DES [128/128 BS SSE2])
guesses: 0  time: 0:00:00:15 DONE (Sat Sep 10 00:02:00 2011)  c/s: 753414  trying:
4bodiadm - 8bodiadm

real    0m15.807s
user    0m15.017s
sys    0m0.260s
```

..and now lets use the MPI setup:

```
# time mpiexec -np 2 ./john --rules --wordlist=/admin/002_dict/001_general/test.dic /
admin/001_to-crack/demo.txt
Loaded 10 password hashes with 10 different salts (Traditional DES [128/128 BS SSE2])
MPI: each node loaded 1/2 of wordfile to memory (about 4693 KB/node)
Node 0 finished at 0:00:00:06.
Node 1 finished at 0:00:00:06.
  0: guesses: 0 time: 0:00:00:06 DONE (Sat Sep 10 00:02:34 2011) c/s: 1013K trying:
08975434 - 08795745
  1: guesses: 0 time: 0:00:00:06 DONE (Sat Sep 10 00:02:34 2011) c/s: 1014K trying:
089818sh - 8bodiadm

real    0m6.612s
user    0m0.072s
sys    0m0.020s
```

Now we can see very specifically that taking just a bit of time, allows us to get JTR to deliver exceptional results.

Final words

There are great tools out there, but taking just a bit of time will allow you to get the best performance out of it. Learn and have fun.