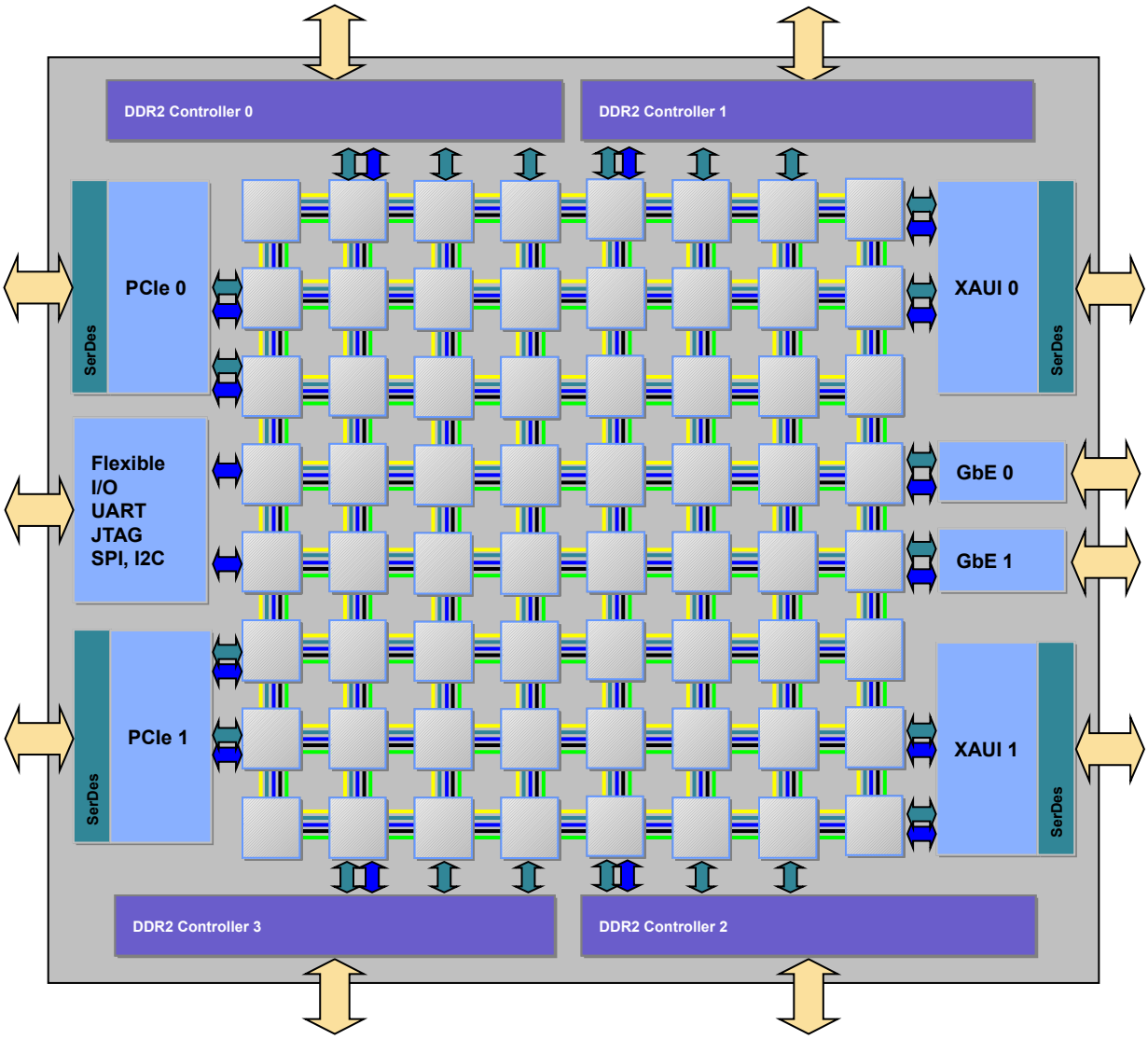
并行运算进入 Tilera 时代

作者：徐鹤军　　单位：上海泛腾电子有限公司

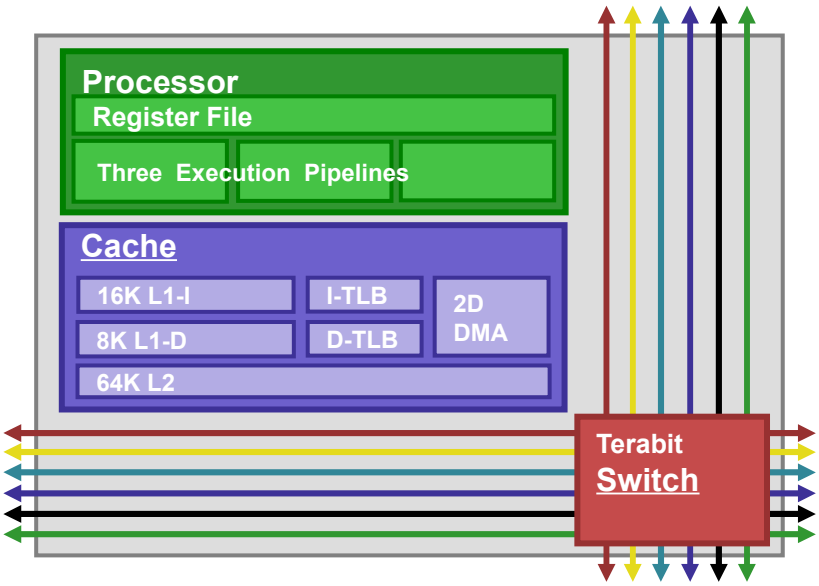
当多数计算机工程师还在为如何充分发挥当前多核 CPU 的实际运行效果而绞尽脑汁时，Tilera 公司内含 64 颗物理计算核的众核 CPU 已经开始的商业应用。这是一款专门为了高性能并行运算而诞生的具有革命性技术革新的 CPU，采用当今最先进的两维 iMesh 构架，完全摒弃了传统多核 CPU 总线互联的结构，完全解决了传统多核 CPU 并行运算时常有的效率瓶颈，也为今后集成更多物理内核做好的基础构架。

下图为 TileraPro64 64 核 CPU 的示意图：



单颗 TileraPro64 CPU 集成了 64 颗 866Mhz 的计算内核，4 个 DDR2 内存控制器，计算内核与内存之间的数据读写从此顺畅无比。尤其还集成了 2 个万兆网络控制器，摒弃了传统构架中的 PCI-E 总线的数据传输瓶颈。从此对于网络数据包能够真正达到了万兆级的处理能力。

每颗物理计算核都配有交换单元与多条总线连接，是内核之间的连接不再有瓶颈，数据交换更加通畅和实时。
下图为 TileraPro64 单颗计算内核的结构示意图：



革命性的 iMesh 构架为 Tilera 众核 CPU 高性能运算作好了技术基础，无怪乎 Tilera 众核 CPU 能比众多竞争者表现出更出色的性能。

下图为 Tilera 众核 CPU 与其他 CPU 的对比图标：

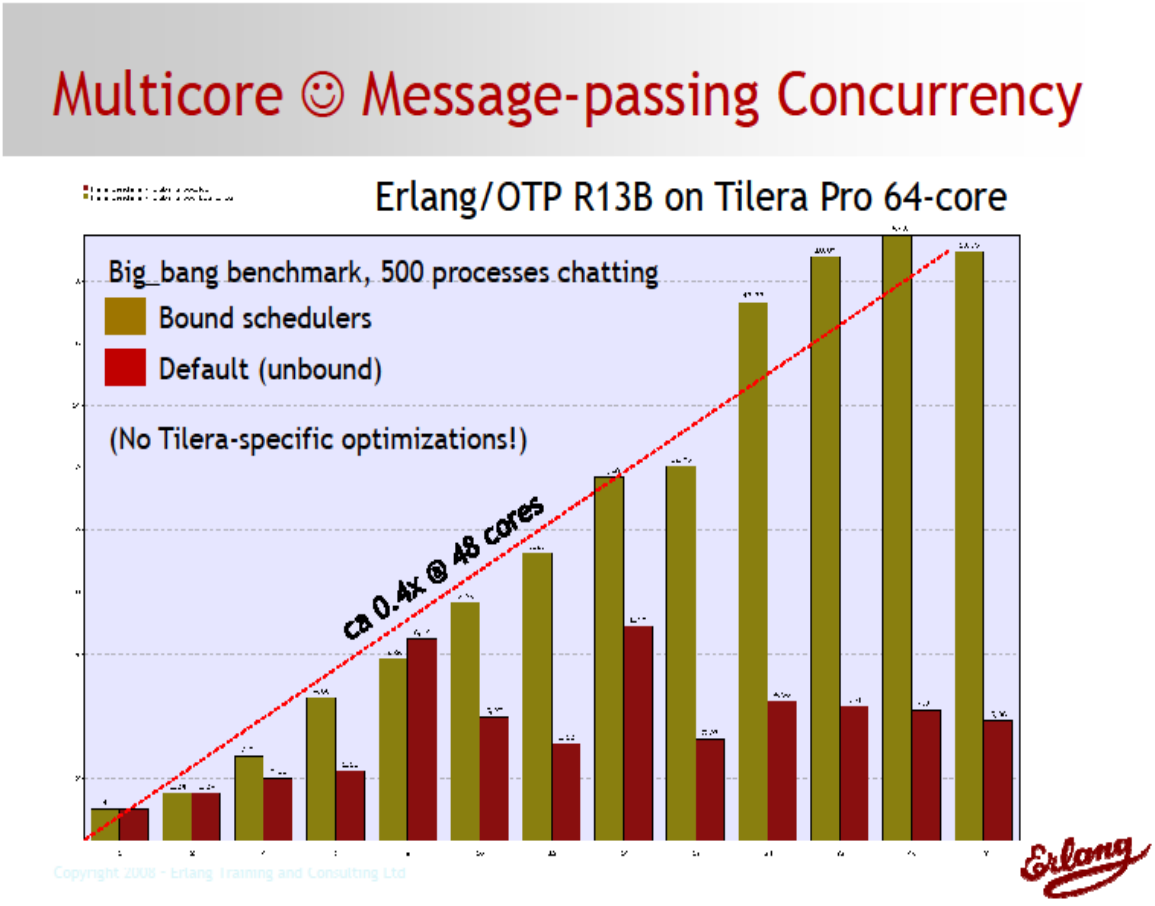
Compare Multi-core Processors

	Clock (MHz)	CPU Power (W)	Chipset Power (W)	Packet I/O (Gb/s)	Mem I/O(Gb/s)	Processor Cores	Core Issue Width	Peak BIPS	Peak BIPS/W
Tilera TILE*Pro*64	700	22	0	20	200	64	3	134.0	6.10
Cisco SPP	250	35	0	192	175	188	1	47.0	1.34
Intel IXP2855	1500	27	0	25	122	16	1	24.0	0.89
Cavium Octeon CN5860	1000	40	0	25	102	16	2	32.0	0.80
Intel Atom Z530, SCH/US15W chipset	1600	2.2	2.3	0	34	1	2	3.2	0.71
Cavium Octeon CN3860	600	30	0	25	102	16	2	19.2	0.64
Intel Xeon 5508, 5520 chipset	2000	38	27.1	410	205	4	4	32.0	0.49
Intel Xeon 5540, 5520 chipset	2530	80	27.1	410	205	4	4	40.5	0.38
Intel Quad-Core Xeon 5300, 5000P chipset	2330	80	30	0	86	4	4	37.3	0.34
AMD Turion 64X2 Dual-Core Mobile TL-56	1800	33	0	51	86	2	3	10.8	0.33
Intel Mobile Core 2Duo, 965e chipset	2400	35	28	0	68	2	4	19.2	0.30
Intel Dual-Core Xeon 5138, Intel 5000P chipset	2130	35	30	0	86	2	4	17.0	0.26
NetLogic XLR 732	1000	32	0	25		8	1	8.0	0.25
AMD Dual-Core Opteron1218 HE	2600	65	0	192	86	2	3	15.6	0.24
Intel Dual-Core Xeon7120, E8501 chipset	3000	96	32	0	51	2	4	24.0	0.19

多 CPU 的编程一直是困扰软件开发人员的技术瓶颈，当前的编程模式难以应对多 CPU 并行运算的效率，往往是随着的更多 CPU 加入运算队列，性能提升不明显甚至不升反降，多 CPU 反而成为累赘。

这时一门叫 Erlang 冷门编程语言开始引起大家的关注，20 年前由 Joe Armstrong 教授为爱立信通信公司的电话交换机系统开发的 Erlang 编程语言，以不同当前编程模式的设计模式，支持超大量级的并发线程。

Erlang 社区的名人 Ulf Wiger 采用 Erlang 语言在 TileraPro64 众核 CPU 上实现了运算性能的线性提升，请看下图标：



基于 Tilera 提供的良好软件开发环境，国内 Erlang 语言的先锋任职阿里巴巴淘宝网的余峰先生也在有上海泛腾电子科技有限公司提供的测试机上轻松实现了 Erlang 语言的编译和运行。

下面是由余峰先生提供的编译过程：

```
# wget http://www.erlang.org/download/otp_src_R13B03.tar.gz  %%R13B04 后 otp_build 变了好像有点问题，编译不过去。
# tar xzf otp_src_R13B03.tar.gz
# cd otp_src_R13B03
%%由于 R13B03 有个小小的 bug 会导致 segment fault,我们需要对
```

erl_misc_utils.c打个patch,以下是要打patch的地方:

```
# diff -Nau
otp_src_R13B04/erts/lib_src/common/erl_misc_utils.c{~,}
--- otp_src_R13B04/erts/lib_src/common/erl_misc_utils.c~
2009-11-20 14:32:23.000000000 +0100
+++ otp_src_R13B04/erts/lib_src/common/erl_misc_utils.c 2010-03-29
15:53:28.000000000 +0200
@@ -373,8 +373,8 @@
       return
0;
    memcpy((void *)
topology,
          (void *) cpuinfo-
>topology,
-           cpuinfo->configured*sizeof(erts_cpu_topology_t));
-    return cpuinfo->configured;
+           cpuinfo->topology_size*sizeof(erts_cpu_topology_t));
+    return cpuinfo->topology_size;
 }
```

%%源码算是准备完毕了

%%开始编译前的环境准备

```
# export ERL_TOP=`pwd`
# eval `./otp_build env_cross $ERL_TOP/xcomp/erl-xcomp-
TileraMDE2.0-tilepro.conf`
# export TILERA_ROOT=/root/TileraMDE-2.1.2.112814/tilepro/  %%这个
最好用2.x版本的
# export PATH=$TILERA_ROOT/bin:$PATH
# tile-monitor --pci -- uname -a  %%验证板卡确实可用
Linux localhost 2.6.26.7-MDE-2.1.2.112814 #1 SMP Wed Sep 1
00:05:06 EDT 2010 tile GNU/Linux
```

%%开始交叉编译

```
# ./otp_build configure
# touch lib/crypto/SKIP &&  touch lib/ssl/SKIP && touch
lib/ssh/SKIP   %%这几个模块用到了openssl，编译不过去，忽略掉
# ./otp_build boot -a
# ./otp_build release -a /tmp/otp %%安装到/tmp/otp目录去
```

%%如果一切顺利，到这时候就编译完毕了。

%%我们开始收获成果。

```
# cd /tmp/otp
# ./Install `pwd`
#  tile-monitor --pci --here -- bin/erl  %%没出错的话，就说明我们成功了。

%%当然我们可以把 otp 系统上载到板卡去，方便日后使用
# $TILERA_ROOT/bin/tile-monitor --pci  --upload /tmp/otp /tmp/otp
--quit   %% 放在板卡的/tmp/otp 目录下

%%我们还可以定义个命令别名，方便我们执行命令
# tile_erl="$TILERA_ROOT/bin/tile-monitor --pci --resume --tunnel
2023 23  --env HOME=/tmp  --tiles - all ^0 - -- /tmp/otp/bin/erl"
# $tile_erl +sct L10-18,1-9,19-55,57,58,61c1-55,57,58,61 +sbt db
-noshell +S 58 -s init stop
```

好吧到此为止，我们在目标机器上/tmp/otp 目录下有个完整的 erlang 系统。

我们开个 console 连接到板载的 linux 系统去：

view source

print?

```
[root@client otp_src_R13B03]# /root/TileraMDE-
2.1.2.112814/tilepro/bin/tile-console
tile-console: Assuming '--pci'.
Connecting to /dev/ttyS0, speed 115200
 Escape character: Ctrl-\ (ASCII 28, FS):
enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
----------------------------------------------------
# uname -a
Linux localhost 2.6.26.7-MDE-2.1.2.112814 #1 SMP Wed Sep 1
00:05:06 EDT 2010 tile GNU/Linux

# cd /tmp/otp
%%我们跑个 erlang 程序<em>64 核心并行</em>进行数学计算
%% 代码在这里下载 http://shootout.alioth.debian.org/u32/program.php?
test=spectralnorm&lang=hipe&id=2
#cat > spectralnorm.erl
%   The Computer Language Benchmarks Game
%   http://shootout.alioth.debian.org/
%   contributed by Fredrik Svahn

-module(spectralnorm).
-export([main/1]).
```

```
-compile( [ inline, { inline_size, 1000 } ] ).

main([Arg]) ->
    register(server,
self()),
    N =
list_to_integer(Arg),
    {U, V} = power_method(N, 10, erlang:make_tuple(N,
1), []),
    io:format("~.9f\n", [ eigen(N, U, V, 0,
0) ]),
    erlang:halt(0
).

% eigenvalue of V
eigen(0, _, _, VBV, VV) when VV /= 0 -> math:sqrt(VBV / VV);

eigen(I, U, V, VBV, VV) when I /= 0 ->
    VI = element(I,
V),
    eigen(I-1, U, V, VBV + element(I, U)*VI, VV
+ VI*VI).

% 2I steps of the power method
power_method(_, 0, A, B) -> {A, B};
power_method(N, I, A, _B) ->
    V = atav(N,
A),
    U = atav(N,
V),
    power_method(N, I-1, U,
V).

% return element i,j of infinite matrix A
a(II,JJ) -> 1/((II+JJ-2)*(II-1+JJ)/2+II).

% multiply vector v by matrix A
av(N, V) -> pmap(N, fun(Begin, End) -> av(N, Begin, End, V) end).

av(N, Begin, End, V) -> server ! { self(), [ avloop(N, I, V, 0.0)
|| I <- lists:seq(Begin, End) ]}.

avloop(0, _, _, X) ->  X;
```

```
avloop(J, I, V, X) ->  avloop(J-1, I, V, X + a(I, J)*element(J, V)
).

% multiply vector v by matrix A transposed
atv(N, V) -> pmap(N, fun(Begin, End)-> atv(N, Begin, End, V) end).

atv(N, Begin, End, V) -> server ! { self(), [ atvloop(N, I, V,
0.0) || I <- lists:seq(Begin, End) ]}.

atvloop(0, _, _, X) -> X;
atvloop(J, I, V, X) -> atvloop(J-1, I, V, X + a(J, I)*element(J,
V) ).

% multiply vector v by matrix A and then by matrix A transposed
atav(N, V) -> atv(N, av(N, V)).

%Helper function for multicore
pmap(N, F) ->
    Chunks = chunks(0,
erlang:system_info(logical_processors), N, []),
    Pids = [spawn(fun()-> F(Begin, End) end) || {Begin, End}
<- Chunks],
    Res = [ receive {Pid, X} -> X end || Pid
<- Pids],
    list_to_tuple(lists:flatten(R
es)).

chunks(I, P, N, A) when I == P-1 -> lists:reverse([{I*(N div P)+1,
N} | A ]);
chunks(I, P, N, A) -> chunks(I+1, P, N, [{ I*(N div P)+1, (I+1)*(N
div P)} | A ]).

CTRL+D

# bin/erlc spectralnorm.erl
# time bin/erl -noshell -run spectralnorm  main 500  -s init stop
1.274224116
real    0m 16.90s
user    1m 16.07s
sys     0m 0.79s
# bin/erl  +sct L10-18,1-9,19-55,57,58,61c1-55,57,58,61 +sbt db
-noshell +S 58 -eval 'io:format("schedulers:
~p~n"[erlang:system_info(schedulers)])' -s init stop
schedulers: 58
```

Bingo! 我们顺利的跑了数学计算，同时绑定调度器到 CPU core 上去了。接下来就可以按照平常那样进行 Erlang 并发编程了。

有关余峰先生的相关文章和资料请看 http://blog.yufeng.info.