TENSORFLOW* ON MODERN INTEL® ARCHITECTURES

Jing Huang and Vivek Rane
Artificial Intelligence Product Group
Intel
TensorFlow* on CPU has been very slow

You must choose one of the following types of TensorFlow to install:

- **TensorFlow with CPU support only.** If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install (typically, in 5 or 10 minutes), so even if you have an NVIDIA GPU, we recommend installing this version first.

- **TensorFlow with GPU support.** TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting the prerequisites shown below and you need to run performance-critical applications, you should ultimately install this version.

https://www.tensorflow.org/install/install_linux

**Until today.**

Up to 72x Speedup in Training and 86x Speedup in Inference! Up-streamed and Ready to Use!
AGENDA

• Deep Learning & TensorFlow
• Optimizing TensorFlow on Intel® Architecture
• Summary & Call to Action
• Tutorial on Cray (cori) systems
Deep Learning: Convolutional Neural Network

Convolution Parameters:
Number of outputs/feature-maps: < 4 >
Filter size: < 3 x 3 >
Stride: < 2 >
Pad_size (for corner case): <1>

Filter = 3 x 3
Stride = 2
Pad_size = 1

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
Deep Learning: Train Once Use Many Times

**Step 1: Training**
(Over Hours/Days/Weeks)

- Input data
- Create deep network
- Output classification

**Input data**

90% person
8% traffic light

**Step 2: Inference**
(Real Time)

- New input from camera and sensors
- Trained neural network model
- Output classification

97% person
Bigger Data

Image: 1000 KB / picture
Audio: 5000 KB / song
Video: 5,000,000 KB / movie

Better Hardware

Transistor density doubles every 18 months
Cost / GB in 1995: $1000.00
Cost / GB in 2015: $0.03

Smarter Algorithms

Advances in algorithm innovation, including neural networks, leading to better accuracy in training models
TensorFlow

• 2nd generation open source machine learning framework from Google*
• Widely used across Google in many key apps – search, Gmail, photos, translate, etc.
• General computing mathematical framework used on:
  • Deep neural network
  • Other machine learning frameworks
  • HPC applications
• Core system provides set of key computational kernel, extendable user ops
• Core in C++, front end wrapper is in python specifies/drives computation
• Multi-node support using proprietary GRPC protocols

*Other names and brands may be claimed as the property of others.
TENSORFLOW: COMPUTATION IS A DATAFLOW GRAPH WITH TENSORS

- Google's open source machine learning framework
- [https://github.com/tensorflow/tensorflow](https://github.com/tensorflow/tensorflow)

Example from Jeff Dean's presentation

Edges are n-dimensional arrays: Tensors
AGENDA

• Deep Learning & TensorFlow

• Optimizing TensorFlow on Intel® Architecture
  • Why Optimize
  • Optimization Challenges & Techniques Used
  • Performance Results

• Summary & Call to Action
OPTIMIZATION MATTERS ON MODERN ARCHITECTURES WITH HIGH CORE COUNTS AND WIDE SIMD VECTORS
MOORE’S LAW GOES ON!

Increasing clock speeds -> more cores + wider SIMD (Hierarchical parallelism)
**COMBINED AMDAHL’S LAW FOR VECTOR MULTICORES**

\[
\text{Speedup} = \left( \frac{1}{\text{Serial}_{\text{frac}} + \frac{1 - \text{Serial}_{\text{frac}}}{\text{NumCores}}} \right) \times \left( \frac{1}{\text{Scalar}_{\text{frac}} + \frac{1 - \text{Scalar}_{\text{frac}}}{\text{VectorLength}}} \right)
\]

Goal: Reduce **Serial Fraction** and Reduce **Scalar Fraction** of Code

**Compute Bound Performance**
Most kernels of ML codes are compute bound i.e. raw FLOPS matter

**Roofline Model**
Gflops/s = min (Peak Gflops/s, Stream BW * flops/byte)
OPTIMIZING TENSORFLOW AND DEEP LEARNING WORKLOADS
Performance Optimization on Modern Platforms

Hierarchical Parallelism

Coarse-Grained / multi-node
Domain decomposition

- Improve load balancing
- Reduce synchronization events, all-to-all comms

Fine-Grained Parallelism / within node
Sub-domain: 1) Multi-level domain decomposition (ex. across layers)
2) Data decomposition (layer parallelism)

Scaling

Utilize all the cores

- OpenMP, MPI, TBB...
- Reduce synchronization events, serial code
- Improve load balancing

Vectorize/SIMD

- Unit strided access per SIMD lane
- High vector efficiency
- Data alignment

Efficient memory/cache use

- Blocking
- Data reuse
- Prefetching
- Memory allocation

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
# Intel Strategy: Optimized Deep Learning Environment

<table>
<thead>
<tr>
<th>Fuel the development of vertical solutions</th>
<th>Accelerate design, training, and deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nervana Cloud™</strong></td>
<td><strong>Drive optimizations across open source machine learning frameworks</strong></td>
</tr>
<tr>
<td>Caffe, Theano, TensorFlow, PyTorch, DMLC, MXNet</td>
<td><strong>Maximum performance on Intel architecture</strong></td>
</tr>
<tr>
<td>Intel® Math Kernel Library (Intel® MKL)</td>
<td>** Deliver best single node and multi-node performance**</td>
</tr>
<tr>
<td>Intel® Nervana™ Graph</td>
<td></td>
</tr>
</tbody>
</table>

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. *
Other names and brands may be claimed as the property of others.
Example Challenge 1: Data Layout Has Big Impact on Performance

- Data layouts impact performance
  - Sequential access to avoid gather/scatter
  - Have iterations in inner most loop to ensure high vector utilization
  - Maximize data reuse; e.g. weights in a convolution layer
- Converting to/from optimized layout is sometimes less expensive than operating on unoptimized layout

Better optimized for some operations vs
EXAMPLE CHALLENGE 2: MINIMIZE CONVERSIONS OVERHEAD

• End-to-end optimization can reduce conversions
• Staying in optimized layout as long as possible becomes one of the tuning goals
• Minimize the number of back and forth conversions
  • Use of graph optimization techniques
OPTIMIZING TENSORFLOW & OTHER DL FRAMEWORKS FOR INTEL® ARCHITECTURE

- Leverage high performant compute libraries and tools
  - e.g. Intel® Math Kernel Library, Intel® Python, Intel® Compiler etc.
- Data format/shape:
  - Right format/shape for max performance: blocking, gather/scatter
- Data layout:
  - Minimize cost of data layout conversions
- Parallelism:
  - Use all cores, eliminate serial sections, load imbalance
- Memory allocation
  - Unique characteristics and ability to reuse buffers
- Data layer optimizations:
  - Parallelization, vectorization, IO
- Optimize hyper parameters:
  - e.g. batch size for more parallelism
  - Learning rate and optimizer to ensure accuracy/convergence
### Initial Performance Gains on Modern Xeon (2 Sockets Broadwell – 22 Cores)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Metric</th>
<th>Batch Size</th>
<th>Baseline Performance Training</th>
<th>Baseline Performance Inference</th>
<th>Optimized Performance Training</th>
<th>Optimized Performance Inference</th>
<th>Speedup Training</th>
<th>Speedup Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet-Alexnet</td>
<td>Images / sec</td>
<td>128</td>
<td>33.52</td>
<td>84.2</td>
<td>524</td>
<td>1696</td>
<td>15.6x</td>
<td>20.2x</td>
</tr>
<tr>
<td>ConvNet-GoogleNet v1</td>
<td>Images / sec</td>
<td>128</td>
<td>16.87</td>
<td>49.9</td>
<td>112.3</td>
<td>439.7</td>
<td>6.7x</td>
<td>8.8x</td>
</tr>
<tr>
<td>ConvNet-VGG</td>
<td>Images / sec</td>
<td>64</td>
<td>8.2</td>
<td>30.7</td>
<td>47.1</td>
<td>151.1</td>
<td>5.7x</td>
<td>4.9x</td>
</tr>
</tbody>
</table>

- Baseline using TensorFlow 1.0 release with standard compiler knobs
- Optimized performance using TensorFlow with Intel optimizations and built with
  - bazel build --config=mkl --copt="-DEIGEN_USE_VML"
# Initial Performance Gains on Modern Xeon Phi (Knights Landing - 68 Cores)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Metric</th>
<th>Batch Size</th>
<th>Baseline Performance Training</th>
<th>Baseline Performance Inference</th>
<th>Optimized Performance Training</th>
<th>Optimized Performance Inference</th>
<th>Speedup Training</th>
<th>Speedup Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet-Alexnet</td>
<td>Images / sec</td>
<td>128</td>
<td>12.21</td>
<td>31.3</td>
<td>549</td>
<td>2698.3</td>
<td>45x</td>
<td>86.2x</td>
</tr>
<tr>
<td>ConvNet-GoogleNet v1</td>
<td>Images / sec</td>
<td>128</td>
<td>5.43</td>
<td>10.9</td>
<td>106</td>
<td>576.6</td>
<td>19.5x</td>
<td>53x</td>
</tr>
<tr>
<td>ConvNet-VGG</td>
<td>Images / sec</td>
<td>64</td>
<td>1.59</td>
<td>24.6</td>
<td>69.4</td>
<td>251</td>
<td>43.6x</td>
<td>10.2x</td>
</tr>
</tbody>
</table>

- Baseline using TensorFlow 1.0 release with standard compiler knobs
- Optimized performance using TensorFlow with Intel optimizations and built with
  - `bazel build --config=mkl --copt="-DEIGEN_USE_VML"`
ADDITONAL PERFORMANCE GAINS FROM PARAMETERS TUNING

- Data format: CPU prefers NCHW data format
- Intra_op, inter_op and OMP_NUM_THREADS: set for best core utilization
- Batch size: higher batch size provides for better parallelism
  - Too high a batch size can increase working set and impact cache/memory perf

Best Setting for Xeon (Broadwell – 2 Socket – 44 Cores)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Data Format</th>
<th>Inter_op</th>
<th>Intra_op</th>
<th>KMP_BLOCKTIME</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet- AlexnetNet</td>
<td>NCHW</td>
<td>1</td>
<td>44</td>
<td>30</td>
<td>2048</td>
</tr>
<tr>
<td>ConvNet-Googlenet V1</td>
<td>NCHW</td>
<td>2</td>
<td>44</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>ConvNet-VGG</td>
<td>NCHW</td>
<td>1</td>
<td>44</td>
<td>1</td>
<td>128</td>
</tr>
</tbody>
</table>

Best Setting for Xeon Phi (Knights Landing – 68 Cores)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Data Format</th>
<th>Inter_op</th>
<th>Intra_op</th>
<th>KMP_BLOCKTIME</th>
<th>OMP_NUM_THREADS</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet- AlexnetNet</td>
<td>NCHW</td>
<td>1</td>
<td>136</td>
<td>30</td>
<td>136</td>
<td>2048</td>
</tr>
<tr>
<td>ConvNet-Googlenet V1</td>
<td>NCHW</td>
<td>2 training</td>
<td>68</td>
<td>Infinite</td>
<td>68</td>
<td>256</td>
</tr>
<tr>
<td>ConvNet-VGG</td>
<td>NCHW</td>
<td>1</td>
<td>136</td>
<td>1</td>
<td>136</td>
<td>128</td>
</tr>
</tbody>
</table>
Optimized Perf: Alexnet on different batch sizes


<table>
<thead>
<tr>
<th>BDW</th>
<th>KNL</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.52</td>
<td>12.21</td>
</tr>
<tr>
<td>524</td>
<td>549</td>
</tr>
<tr>
<td>544</td>
<td>694</td>
</tr>
<tr>
<td>563</td>
<td>764</td>
</tr>
<tr>
<td>566</td>
<td>839</td>
</tr>
<tr>
<td>570 (17x)</td>
<td>872 (71.5x)</td>
</tr>
</tbody>
</table>

72x Speedup From New Optimizations – available through Google's TensorFlow Git
PERFORMANCE GAINS – CONVNET-GOOGLENET V1 TRAINING (IMAGES PER SECOND)

Optimized Perf: Googlenet on different batch sizes

Baseline Perf Training | Optimized Perf Training - batch size 128 | Optimized Perf Training - batch size 256

BDW | KNL

16.8 | 5.4 | 140 (26x)
112 | 106 | 113 (6.7x)

26x Speedup From New Optimizations – available through Google’s TensorFlow Git
PERFORMANCE GAINS – CONVNET-vgg training (images per second)

Optimized Perf: VGG on different batch sizes

Baseline Perf Training | Optimzed Perf Training - batch size 64 | Optimzed Perf Training - batch size 128
-----------------------|----------------------------------------|----------------------------------------
8.2                   | 47                                    | 75 (47x)                               
1.6                   | 1.6                                    | 49 (6x)                                 

47x Speedup From New Optimizations – available through Google’s TensorFlow Git
How Do I Get Order of Magnitude CPU Speedup for My Topology?

- Optimized TensorFlow on Intel architectures available from the public git.
  - git clone https://github.com/tensorflow/tensorflow.git
- Configure for best performance on CPU:
  - Run "./configure" from the TensorFlow source directory
  - Select option for MKL (CPU) optimization
  - Automatically downloads latest MKL-ML
- Building for best performance on CPU
  - Use following command to create a pip package that can be used to install the optimized TensorFlow wheel
    - bazel build --config=mkl --copt="-DEIGEN_USE_VML" --s --c opt //tensorflow/tools/pip_package:build_pip_package
  - Install the optimized TensorFlow wheel
    - bazel-bin/tensorflow/tools/pip_package/build_pip_package ~/path_to_save_wheel
    - pip install --upgrade --user ~/path_to_save_wheel/wheel_name.whl
Maximum performance requires using all the available cores efficiently

Users and data scientists should experiment with environment variable settings

Best setting depend on topology and platform (e.g., number of cores)

Example of ConvNet-Alexnet environment settings on Knights Landing

- KMP_BLOCKTIME = 30
- KMP_SETTINGS = 1
- KMP_AFFINITY = granularity=fine,verbose,compact,1,0
- OMP_NUM_THREADS = 136 (Xeon Phi has 68 physical cores)

Knobs in the Python topology can also impact performance:

- Data format: using NCHW format to avoid additional internal format conversions to get maximum performance
- Matmul layer: the second input matrix should be transposed for better performance
- Intra_op /inter_op: experiment with intra_op/inter_op for each topology/ platform

Example of ConvNet-Alenet settings on Xeon Phi

- inter_op = 2
- intra_op = 136
• TensorFlow is widely used DL and AI framework
  • It has been slow on CPU until now
• Significant performance gains from optimization on modern Intel® Xeon® and Xeon Phi™ processors
• Traditional optimization techniques: vectorization, parallelization, cache blocking, etc.
• Unique performance challenges: data layout, hyper parameters, inter/intra layer parallelization, etc.
CALL TO ACTION

Latest TensorFlow with Intel optimizations directly from TensorFlow GIT repository

Use the right configuration, building and best parameter settings

Orders of magnitude higher CPU performance for inference and training
1. Get TensorFlow from Google's repository
   git clone https://github.com/tensorflow/tensorflow.git

2. Get convnet Alexnet

3. Load Java (bazel 0.5.4 needs Java 1.8+)
   module load java

4. Load Python/Pip
   module load python

5. Load gcc (need 5.4+)
   module load gcc

6. Setup Bazel
   wget https://github.com/bazelbuild/bazel/releases/download/0.4.5/bazel-0.4.5-installer-linux-x86_64.sh
   chmod +x bazel-0.4.5-installer-linux-x86_64.sh
   ./bazel-0.4.5-installer-linux-x86_64.sh --user export PATH=~/bin/:$PATH

7. Configure tensorflow
   cd tensorflow
   ./configure
   MKL -> yes
   everything else -> default

8. Build!
   bazel build --config=mkl --copt="-DEIGEN_USE_VML" -s -c opt

9. Make a wheel
   bazel-bin/tensorflow/tools/pip_package/build_pip_package
   //tensorflow/tools/pip_package

10. Install the wheel
    (If you don't want to build, there is a pre-built one using the instructions above available at:
        /global/cscratch1/sd/vrane/tensorflow-1.1.0-cp27-cp27mu-linux_x86_64.whl)

11. Set PYTHONPATH to installed location
    export PYTHONPATH=/<full-path>/install/

12. Find a node to run the benchmark
    salloc --reservation=CUG2C -N 1 -p regular -C knl,quad,flat -t 60 -A ntrain

13. Run the benchmark you downloaded in step 2 (convnet Alexnet)
    python benchmark_alexnet.py

14. Now optimize the benchmark for KNL:
    a. OMP_NUM_THREADS and inter/intra-op settings
       import os
       os.environ["OMP_NUM_THREADS"] = "136"
       os.environ["KMP_BLOCKTIME"] = "30"
       os.environ["KMP_SETTINGS"] = "1"
       os.environ["KMP_AFFINITY"] = "granularity=fine,verbose,compact,1,0"

    b. Change batch size to 2048
    c. Instead of relu_layer, use matmul (transposed) and relu. This formats the data in a manner that allows for faster processing in MKL.
    d. Set the allocator to BFC, and supply the intra and inter-op parallelism flags to the session.

15. Rerun the benchmark to see the performance improvement.
    [You can find an optimized version of the benchmark here (with the modifications from step 14):
     /global/cscratch1/sd/vrane/benchmark_alexnet_knl.py]
    python benchmark_alexnet.py
MORE FROM INTEL...

**Intel® Nervana™ AI Academy**

Visit: software.intel.com/ai/academy

---

**Kaggle Competition: Intel® & MobileODT Cervical Cancer Screening**

Register: kaggle.com/c/intel-mobileodt-cervical-cancer-screening

---

**Additional Deep Learning Resources**

Visit: nervanasys.com/learn

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
Legal Disclaimers

• Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number

• Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

• Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

• Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.

• Relative performance is calculated by assigning a baseline value of 1.0 to one benchmark result, and then dividing the actual benchmark result for the baseline platform into each of the specific benchmark results of each of the other platforms, and assigning them a relative performance number that correlates with the performance improvements reported.

• SPEC, SPECint, SPECfp, SPECrate, SPECpower, SPECjbb, SPECompG, SPEC MPI, and SPECjEnterprise* are trademarks of the Standard Performance Evaluation Corporation. See http://www.spec.org for more information.


• No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Xeon® processor E7-8800/4800/2800 v2 product families or Intel® Itanium® 9500 series-based system (or follow-on generations of either.) Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

For systems also featuring Resilient System Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

For systems also featuring Resilient Memory Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel® processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
Q&A:
TENSORFLOW® ON MODERN INTEL® ARCHITECTURES
CONFIGURATION DETAILS

Xeon-Broadwell: Intel® Xeon™ processor E5-2699v4 (22 Cores, 2.2 GHz), 128GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2

Xeon Phi – Knights Landing: Intel® Xeon Phi™ processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM: Flat mode), 96GB DDR memory, Centos 7.2 based on Red Hat* Enterprise Linux 7.2

AlexNet, GoogleNet v1 and VGG benchmarks:

https://github.com/soumith/convnet-benchmarks
Latest released – Broadwell (14nm process)

- Intel's foundation of HPC and ML performance
- Suited for full scope of workloads
- Industry leading performance/watt for serial & highly parallel workloads.
- Upto 22 cores / socket (Broadwell-EP) (w/ Hyper-Threading technology)

Software optimization helps maximize benefit and adoption of new features
# INTEL® AVX TECHNOLOGY

## SNB/IVB
- 256b AVX1
  - Flops/Cycle: 16 SP / 8 DP

## HSW/BDW
- 256b AVX2
  - Flops/Cycle: 32SP / 16 DP (FMA)

## SKX & KNL
- 512b AVX512
  - Flops/Cycle: 64SP / 32 DP (FMA)

### AVX
- 256-bit basic FP
- 16 registers
- NDS (and AVX128)
- Improved blend
- MASKMOV
- Implicit unaligned

### AVX2
- Float16 (IVB 2012)
- 256-bit FP FMA
- 256-bit integer
- PERMD
- Gather

### AVX512
- 512-bit FP/Integer
- 32 registers
- 8 mask registers
- Embedded rounding
- Embedded broadcast
- Scalar/SSE/AVX “promotions”
- Native media additions
- HPC additions
- Transcendental support
- Gather/Scatter

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804