■続・clang の CUDA 対応について調査

Diary/2011-3-25 の続き.
/tools/clang/lib/Sema/SemaExpr.cpp の Sema::ActOnCUDAExecConfigExpr の
第 3 引数 MultiExprArg execConfig をみてみる.
MultiExprArg は，tools/clang/include/clang/Sema/Ownership.h で,

```
typedef ASTMultiPtr<Expr*> MultiExprArg;
```

として与えられている型.

```
hoge<<<(512), 1>>>(id, Cd);
```

とかだと，execConfig から,

```
execConfig.size()
```

の値は，2 で

```
execConfig.get()[0]->dump();
execConfig.get()[1]->dump();
```

とかすると,

```
(ParenExpr 0x5621eb8 'int'
  (IntegerLiteral 0x5621e90 'int' 512))
(IntegerLiteral 0x5621ed8 'int' 1)
```

と得られる．ここまでは，まだ値保存されているね，と一安心.
Sema::ActOnCallExpr には第 4 引数の MultiExprArg args で引き渡される.
途中で,

```
Expr **Args = args.release();
```

とかってなってる．release の定義は tools/clang/include/clang/Sema/Ownership.h に

```
  PtrTy *release() {
    return Nodes;
  }
```

とある．Node(Expr 型) へのポインタだけを返すようだ．で,

```
return BuildResolvedCallExpr(Fn, NDecl, LParenLoc, Args, NumArgs, RParenLoc,
                             ExecConfig);
```

なので，第 4 引数で渡される．なので，Sema::BuildResolvedCallExpr で,

```
  Args[0]->dump();
  Args[1]->dump();
```

としてみてみると,

```
(DeclRefExpr 0x42e12b0 'int' lvalue ParmVar 0x42dd110 'id' 'int')
(DeclRefExpr 0x42e12d8 'float *' lvalue Var 0x42dd2d0 'Cd' 'float *')
```

... あれ？

```
Expr **Args = args.release();
```

の直後で，Args を dump してみることにする．もちろん，ここは問題ない．
ん？やはり，最後の return BuildResolvedCallExpr(...) で返っているわけじゃないのか？
いや，return BuildResolvedCallExpr(...) で返っているのは確かだけど,

```
if (Config) {
  TheCall = new (Context) CUDAKernelCallExpr(Context, Fn,
                                             cast<CallExpr>(Config),
                                             Args, NumArgs,
                                             Context.BoolTy,
                                             VK_RValue,
                                             RParenLoc);
} else {
  TheCall = new (Context) CallExpr(Context, Fn,
                                   Args, NumArgs,
                                   Context.BoolTy,
                                   VK_RValue,
                                   RParenLoc);
}
```

ではないみたいだ．フックのいれかたが不適切だったなあ．
というわけで, Config の値をみてみると 0 なので, else 節の方へマッチして, CallExpr が呼ばれている．

```
if (const FunctionProtoType *Proto = dyn_cast<FunctionProtoType>(FuncT)) {
  if (ConvertArgumentsForCall(TheCall, Fn, FDecl, Proto, Args, NumArgs,
                              RParenLoc))
    return ExprError();
} else {
  assert(isa<FunctionNoProtoType>(FuncT) && "Unknown FunctionType!");
```

の then 節にマッチしているようだ．

```
(CXXMethodDecl *Method = dyn_cast_or_null<CXXMethodDecl>(FDecl)) = 0
NDecl = 0x3defb50
FDecl = 0x3defb50
```

だけど，結局最後の

```
return MaybeBindToTemporary(TheCall);
```

のとこで返るみたい．
TheCall は,

```
(CallExpr 0x4e42f70 '_Bool'
     (ImplicitCastExpr   0 x4 e42 f58   'cudaError_t   (*)(dim3 ,   dim3 ,   size_t,   cudaStream_t)'
<FunctionToPointerDecay>
     (DeclRefExpr 0x4e42f30 'cudaError_t (dim3, dim3, size_t, cudaStream_t)' lvalue Function 0x43f3
b50 'cudaConfigureCall' 'cudaError_t (dim3, dim3, size_t, cudaStream_t)'))
```

```
    (ParenExpr 0x4e42ee8 'int'
      (IntegerLiteral 0x4e42ec0 'int' 512))
    (IntegerLiteral 0x4e42f08 'int' 1))
```

なのだけど，返るところの直前では，

```
 (CallExpr 0x4e42f70 'cudaError_t':'enum cudaError'
      (ImplicitCastExpr   0 x4 e42 f58   'cudaError_t   (*)(dim3 ,   dim3 ,   size_t,   cudaStream_t)'
<FunctionToPointerDecay>
     (DeclRefExpr 0x4e42f30 'cudaError_t (dim3, dim3, size_t, cudaStream_t)' lvalue Function 0x43f3
b50 'cudaConfigureCall' 'cudaError_t (dim3, dim3, size_t, cudaStream_t)'))
   (CXXConstructExpr 0x4e430a8 'dim3':'struct dim3''void (const struct dim3 &) throw()' elidable
     (ImplicitCastExpr 0x4e43090 'const struct dim3' <NoOp>
       (ImplicitCastExpr 0x4e43078 'dim3':'struct dim3' <ConstructorConversion>
          (CXXConstructExpr 0x4e43028 'dim3':'struct dim3''void (unsigned int, unsigned int, unsigned
int)'
            (ImplicitCastExpr 0x4e42fd0 'unsigned int' <IntegralCast>
              (ParenExpr 0x4e42ee8 'int'
                 (IntegerLiteral 0x4e42ec0 'int' 512)))
            (CXXDefaultArgExpr 0x4e42fe8 'unsigned int')
            (CXXDefaultArgExpr 0x4e43008 'unsigned int')))))
   (CXXConstructExpr 0x4e431c0 'dim3':'struct dim3''void (const struct dim3 &) throw()' elidable
     (ImplicitCastExpr 0x4e431a8 'const struct dim3' <NoOp>
       (ImplicitCastExpr 0x4e43190 'dim3':'struct dim3' <ConstructorConversion>
          (CXXConstructExpr 0x4e43140 'dim3':'struct dim3''void (unsigned int, unsigned int, unsigned
int)'
            (ImplicitCastExpr 0x4e430e8 'unsigned int' <IntegralCast>
              (IntegerLiteral 0x4e42f08 'int' 1))
            (CXXDefaultArgExpr 0x4e43100 'unsigned int')
            (CXXDefaultArgExpr 0x4e43120 'unsigned int')))))
   (CXXDefaultArgExpr 0x4e43200 'size_t':'unsigned long')
   (CXXDefaultArgExpr 0x4e43220 'cudaStream_t':'struct CUstream_st *'))
```

となっている．この間に，MaybeBindTemporary が何ども呼び出されている．
dim3 で，512 とか，1 とか，それぞれちゃんと残っているけど，何の関数なんだ，これ？
あらためて，.ll をみてみると，

```
  %tmp78 = load i32* %id.addr, align 4
  %tmp79 = load float** %Cd, align 8
  call void @_Z6hogeiPf(i32 %tmp78, float* %tmp79)
```

とかってなっているから，やっぱりどっかで，値なくなってるんだよなあ ...

```
ExprResult Sema::MaybeBindToTemporary(Expr *E);
```

は include/clang/Sema/Sema.h に定義があって，lib/Sema/SemaExprCXX.cpp に実装がある．

```
  const RecordType *RT = E->getType()->getAs<RecordType>();
  if (!RT)
    return Owned(E);
```

のところでマッチして返るみたい．たしかに RecordType ではないよなあ．
Owned は，

```
  ExprResult Owned(Expr* E) { return E; }
```

は include/clang/Sema/Sema.h に．ちなみに ExprResult の定義は, include/clang/Sema/Ownership.h に．

```
typedef ActionResult<Expr*> ExprResult;
```

なるほど．cudaConfigureCallっていう関数呼び出しになるのか ...
いづれにしても，でてこないけど．