IT4Innovations national supercomputing center

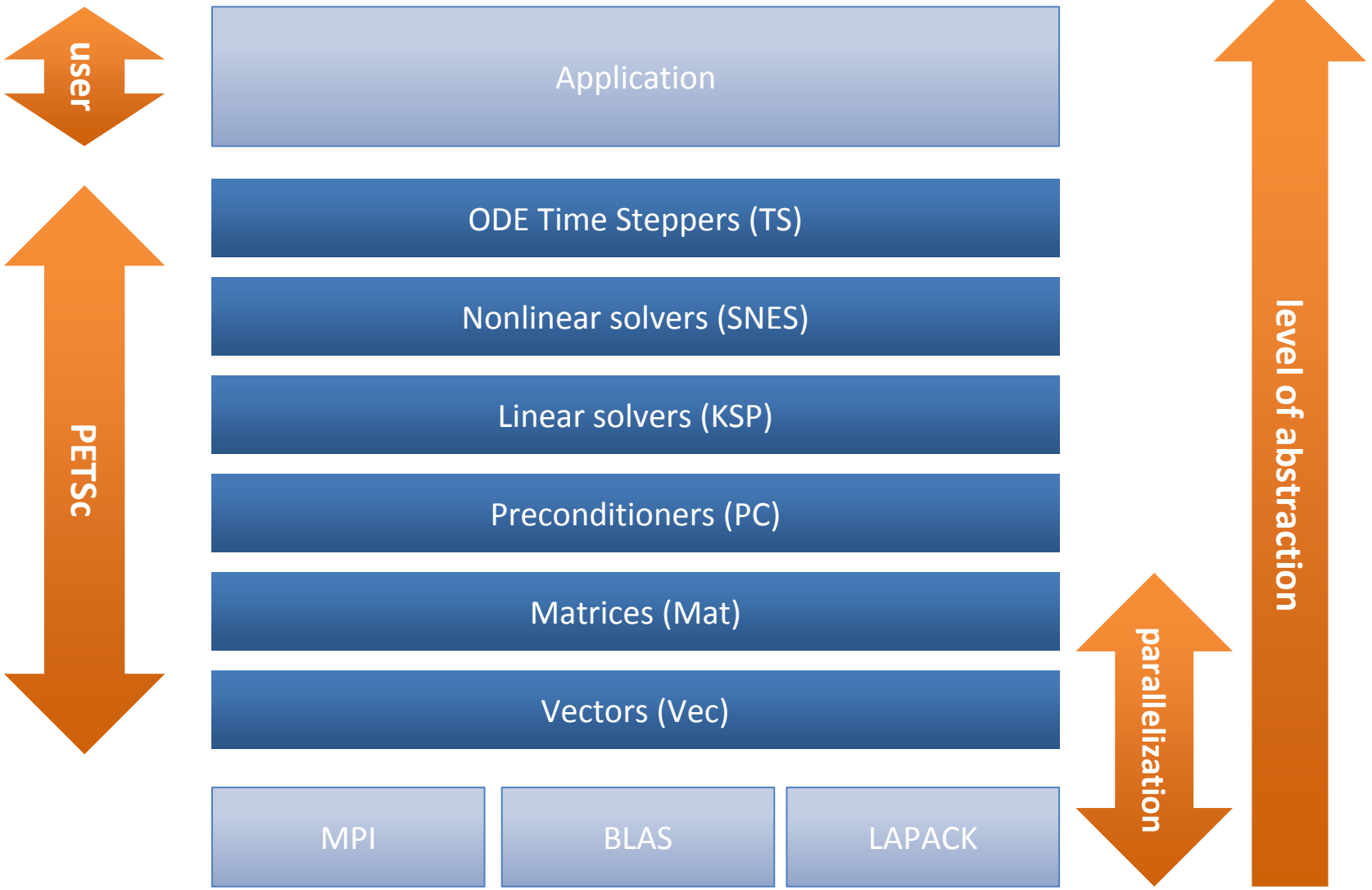
PRACE AUTUMN SCHOOL 2016
September 27-30, 2016
Hagenberg, Austria

PRACE Autumn School 2016
PETSc tutorial

Part III: Objects

Václav Hapla
IT4Innovations
VSB– Technical University of Ostrava
Ostrava, Czech Republic

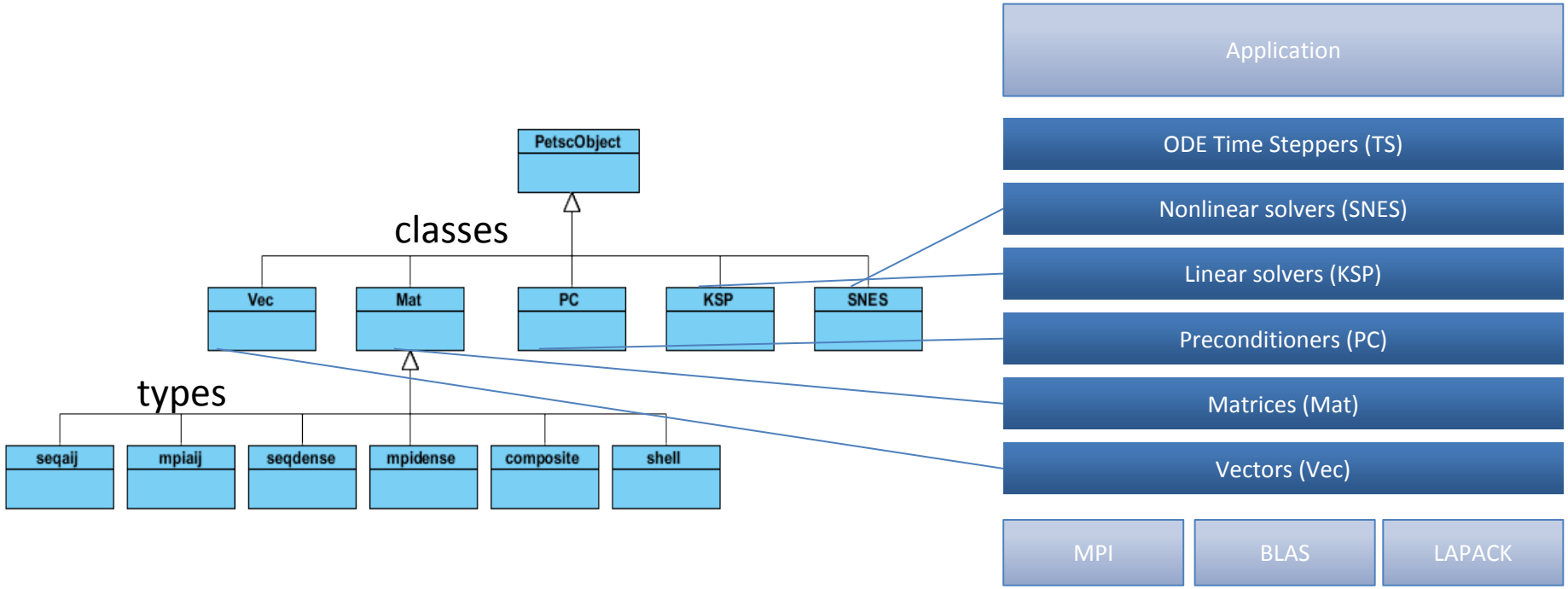
Hierarchy of components



PETSc inheritance

- PETSc uses **3-level inheritance**
- every **object** in PETSc is an instance of a **class**:
  `Vec, Mat, PC, KSP, SNES, …`
- all classes inherit from `PetscObject`
- functions called on objects (methods) are **prefixed** with a **class name**:
  `MatMult(Mat,…)`
- a new object is created with a class-specific **Create** function (constructor):
  `Mat A;  MatCreate(comm, &A);`
- every class is further refined to **types** specified with `SetType`
  `MatSetType(A, MATSEQAIJ);`

PETSc inheritance



PETSc object oriented design: opaque objects

`Mat A,B; Vec x; KSP solver;`

are **opaque objects –** you **don‘t access inner fields directly**

- in `include/petscmat.h` you can find
  `typedef struct _p_Mat* Mat;`
- so `B = A` only **copies pointer**, not data
  - prevents unwanted data copying
  - makes pointer handling easier
  - allows **hiding implementation** from **public interface**
    → **polymorphism**

Polymorphism

```
MatMult(Mat A,Vec x,Vec y);   //y = A*x
```

- **public** interface
  - **uniform** for all types of matrices: sequential, parallel, dense, sparse, blocked, ...
  - **documented**
- calls **private** implementation based on type, e.g. `MatMult_SeqDense(Mat A,Vec x,Vec y)`
  - **specific** for each matrix type (here `MATSEQDENSE`)
  - **not declared** in any header
  - accessible only via `extern` (don't do that)

PetscObject (1)

- Every PETSc object can be cast to `PetscObject`:

```
Mat A;
PetscObject obj;
MatCreate(PETSC_COMM_WORLD, &A);
obj = (PetscObject) A;
```

- `PetscObject` provides general methods such as:
  - `Get/SetName()` – name the object (used for printing, MATLAB interface, etc.)
  - `GetType()` – the type of the object
  - `GetComm()` – the communicator the object belongs to

PetscObject (2)

```
Mat A;
char *type;
MPI_Comm comm;
PetscObjectGetComm((PetscObject)A,&comm);
PetscObjectGetType((PetscObject)A,&type);
   //is the same as
MatGetType(A,&type);
```

Common methods (1)

- once again: **method** names are **prefixed** by the **class** name: `Vec, Mat, KSP, ...`
- all PETSc built-in **classes support** following **methods**

- `Create()` - create the object
- `Get/SetType()` - set the implementation type
- `Destroy()` - deallocate the memory used by the object

Common methods (2)

- `SetFromOptions()` - set all options of the object from the options database
- `Get/SetOptionsPrefix()` - set a specific option prefix for the given object
- `SetUp()` - prepare the object inner state for computation
- `View()` - print object info to specified output

Objects and communicators

- every object in PETSc belongs to some **communicator**
- `MPI_Comm` is the first argument of every object's **constructor**
- two objects can often **interact** only if they belong to the **same communicator** (e.g. `Mat` and `Vec` in matrix-vector product)

Viewers (1)

- `PetscViewer` class will be our first PETSc class
- it is used for printing to **stdout**, **files** (several **text** and **binary** formats), **strings** or even **socket** connection
- **basic usage**:

```
PetscViewer    viewer;

PetscViewerCreate(comm, &viewer);

PetscViewerSetType(viewer, PETSCVIEWERASCII);

PetscViewerDestroy(&viewer);
```

- prints **only from the first processor** of `comm`

Viewers (2)

- **predefined** viewers:

```
PETSC_VIEWER_STDOUT_WORLD, PETSC_VIEWER_BINARY_SELF, ...
```

- **every** PETSc **object** can be **viewed** by the viewer:

```
Viewer v; Mat A; Vec x;
...
MatView(A,v);
VecView(x,v);
```

Hands-on: ex2.c

1. `cd ~/petsc/exercises`
2. `make ex2`
3. `mpirun -n 3 ./ex2`
4. write to file `test.txt` using `PetscViewerFileSetName`
5. set the file name from command line using option `-f`
6. What can be changed with `-n` option? What's the default?
7. What's the expected output of `PetscViewerASCIISynchronizedPrintf`? What's wrong? Fix it!
   (use `PetscViewerFlush`)

Thanks for your attention.