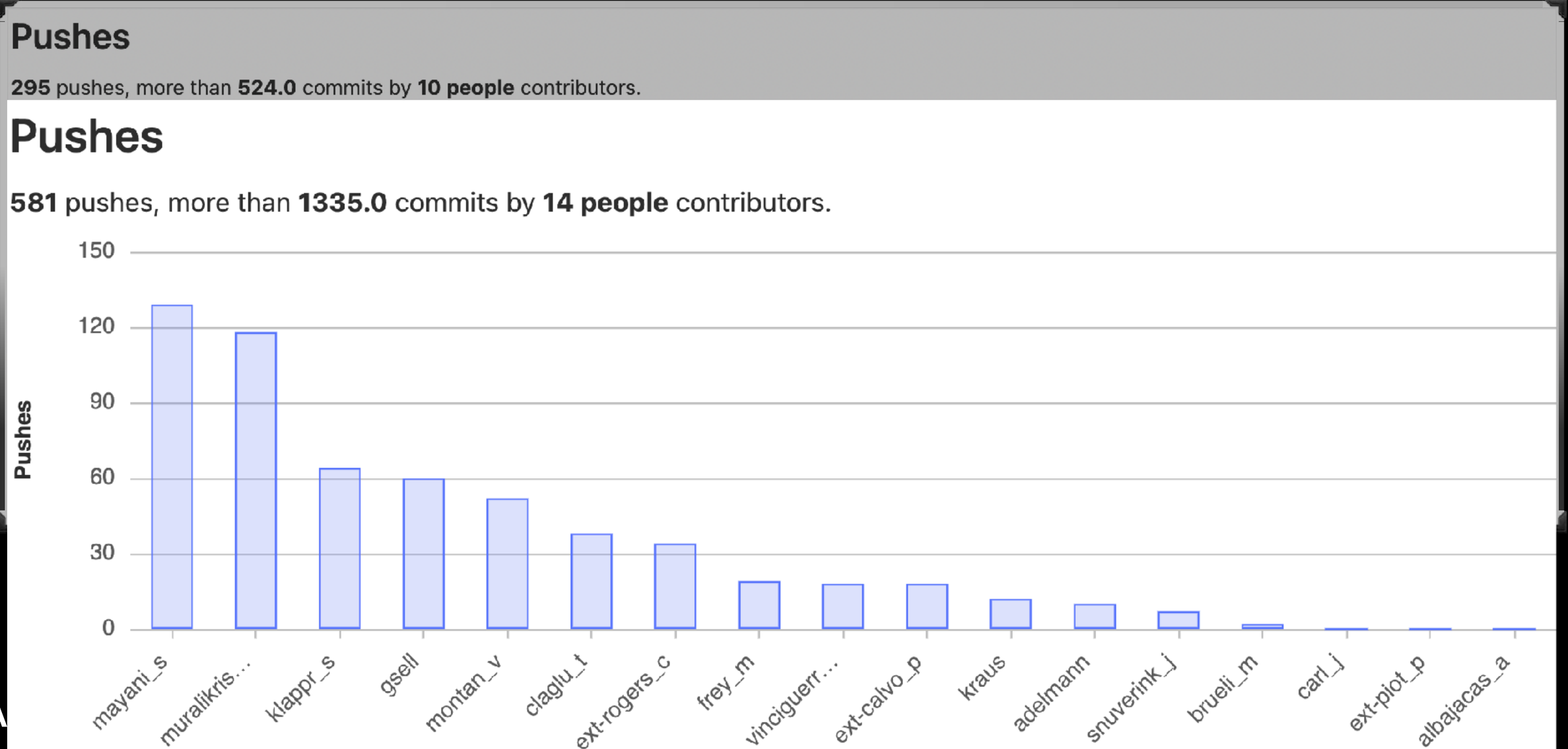
Virtual OPAL Developer Meeting 2023 March - Introduction

https://gitlab.psi.ch/groups/OPAL/-/contribution_analytics?start_date=2022-09-15

Pushes

295 pushes, more than **524.0** commits by **10 people** contributors.



Content

1. Updates OPAL (A Adelmann)
2. Updates on IPPL V 2.x (S Muralikrishnan)
3. Towards OPAL & IPPL V 2.0
4. pyOPAL (Ch Rogers)
5. Next releases (A Gsell)
6. Pressing open issues (all)

Updates

- ~~101~~ 101 member in the active mailing list (opal@lists.psi.ch)
- 1 IPAC contribution (Carl)
- ICFA BD Letter (JInst)

Computational Models for High-Power Cyclotrons and FFAs

Authors	Andreas Adelmann, Chris T Rogers
Publication date	2023/1/4
Journal	arXiv preprint arXiv:2301.01460
Description	A summary of numerical modeling capabilities regarding high power cyclotrons and fixed field alternating gradient machines is presented. This paper focuses on techniques made available by the OPAL simulation code.
Scholar articles	Computational Models for High-Power Cyclotrons and FFAs A Adelmann, CT Rogers - arXiv preprint arXiv:2301.01460, 2023 Related articles All 2 versions

Dr. Moshen Sadar

1.7.2023 + 2 yr

main goal: make OPAL performance portable

Maybe even more good news …

- ISIS has the possibility to bid into a UK fund to get resources for partnership working with PSI
- This would be under the recent MoU that PSI and STFC have just signed
- It will be a bid into UKRI's International Strategic Priority Fund Wave 3 – it will need to get through several selection stages and will be in competition with ideas from other UK research councils
- Funding would most likely be from 24/25 onwards if it is supported

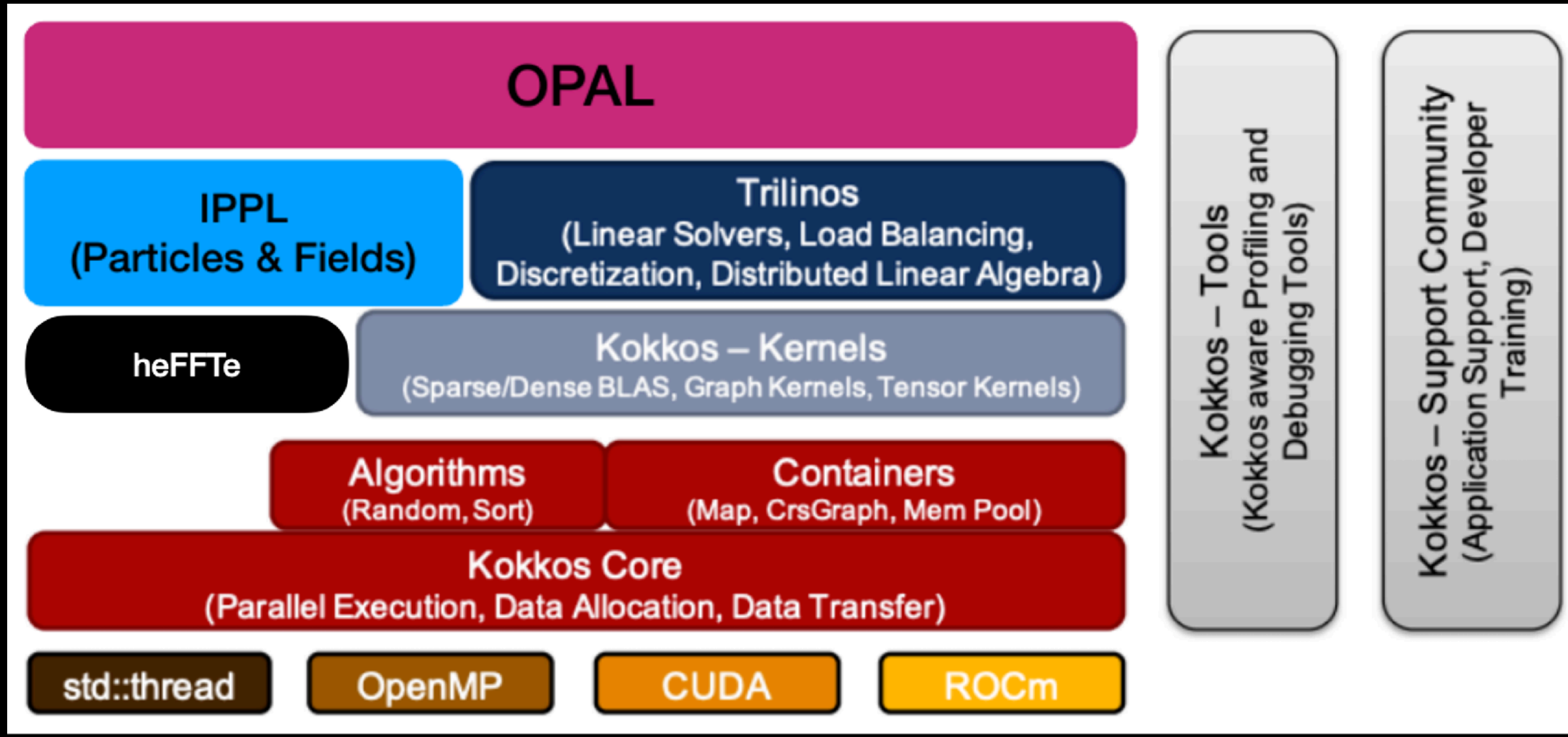
ISIS-PSI Possibilities

Accelerator code development: Joint development of the object-oriented particle accelerator library (OPAL).

Content

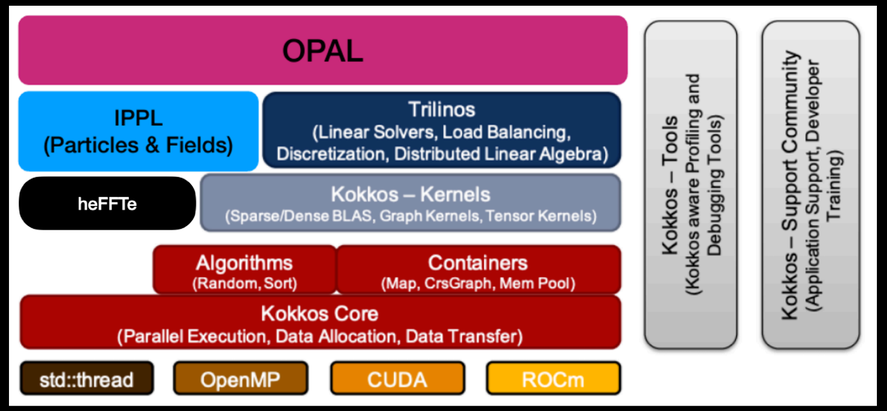
1. Updates OPAL (A Adelmann)
2. Updates on IPPL V 2.x (S Muralikrishnan)
3. Towards OPAL & IPPL V 2.0 (A Adelmann)
4. pyOPAL (Ch Rogers)
5. Next releases (A Gsell)
6. Pressing open issues (all)

Towards OPAL & IPPL V 2.0



Towards OPAL & IPPL V 2.0

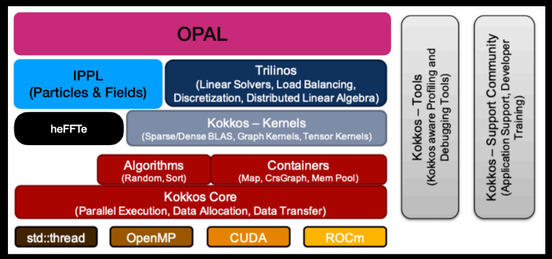
- Separation of OPAL & IPPL
- Make a bare bone OPAL (-t) version
  - DISTRIBUTION, DRIFT, MARKER, MONITOR
- Design choices
  - Solvers in IPPL
  - need to redesign ALL particle loop over fields
    - no if’s, maybe no getField(x,y,z,t) calls
    - host/device
  - OAPL-cycl / OPAL-t unification —> OPAL



```
while (!stepSizes_m.reachedEnd()) {
        unsigned long long trackSteps = stepSizes_m.getNumSteps() + step;
        dtCurrentTrack_m = stepSizes_m.getdT();
        changeDT(back_track);

        for (; step < trackSteps; ++ step) {


            if (itsBunch_m->getTotalNum() > 0) {
                itsBunch_m->get_bounds(rmin, rmax);
            }

            timeIntegration1(pusher);

            itsBunch_m->Ef = Vector_t(0.0);
            itsBunch_m->Bf = Vector_t(0.0);

            computeSpaceChargeFields(step);

            selectDT(back_track);
            emitParticles(step);
            selectDT(back_track);

            computeExternalFields(oth);

            timeIntegration2(pusher);

            itsBunch_m->incrementT();

            if (itsBunch_m->getT() > 0.0 ||
                itsBunch_m->getdT() < 0.0) {
                updateReference(pusher);
            }

            if (deletedParticles_m) {
                evenlyDistributeParticles();
                deletedParticles_m = false;
            }

            itsBunch_m->set_sPos(pathLength_m);

            if (hasEndOfLineReached(globalBoundingBox)) break;

            bool const psDump = ((itsBunch_m->getGlobalTrackStep() % Options::psDumpFreq) + 1 == Options::psDumpFreq);
            bool const statDump = ((itsBunch_m->getGlobalTrackStep() % Options::statDumpFreq) + 1 == Options::statDumpFreq);
            dumpStats(step, psDump, statDump);

            itsBunch_m->incTrackSteps();

            double beta = euclidean_norm(itsBunch_m->RefPartP_m / Util::getGamma(itsBunch_m->RefPartP_m));
            double driftPerTimeStep = std::abs(itsBunch_m->getdT()) * Physics::c * beta;

            if (std::abs(stepSizes_m.getZStop() - pathLength_m) < 0.5 * driftPerTimeStep) {
                break;
            }
        }

        if (globalEOL_m)
            break;

        ++ stepSizes_m;
    }
```

```
for (unsigned int it = 0; it < nt; it++) {

        auto Rview = P->R.getView(); auto Pview = P->P.getView(); auto Eview = P->E.getView(); double V0  = 30 * rmax[2];
        Kokkos::parallel_for(
            "Kick1", P->getLocalNum(), KOKKOS_LAMBDA(const size_t j) {
                double Eext_x = -(Rview(j)[0] - 0.5 * rmax[0]) * (V0 / (2 * std::pow(rmax[2], 2)));
                double Eext_y = -(Rview(j)[1] - 0.5 * rmax[1]) * (V0 / (2 * std::pow(rmax[2], 2)));
                double Eext_z = (Rview(j)[2] - 0.5 * rmax[2]) * (V0 / (std::pow(rmax[2], 2)));

                Eview(j)[0] += Eext_x; Eview(j)[1] += Eext_y; Eview(j)[2] += Eext_z;

                Pview(j)[0] += alpha * (Eview(j)[0] + Pview(j)[1] * Bext);
                Pview(j)[1] += alpha * (Eview(j)[1] - Pview(j)[0] * Bext);
                Pview(j)[2] += alpha * Eview(j)[2];
            });

        P->R = P->R + dt * P->P;
        PL.update(*P, bunchBuffer);

        // will not work !!
        if (P->balance(totalP, it + 1)) {
            P->repartition(FL, mesh, bunchBuffer, isFirstRepartition);
        }

        P->scatterCIC(totalP, it + 1, hr);
        P->solver_mp->solve();
        P->gatherCIC();

…
```

Timeline



- Separation of OPAL & IPPL (AA)
  - Make a bare bone OPAL (-t) version (AA)
  - Moshen intro to OPAL: build, code std, C++, work on some issues (Achim & AA)
  - Moshen intro to IPPL (Sri @ Juelich 2 weeks)

July August September

Content

1. Updates OPAL (A Adelmann)
2. Updates on IPPL V 2.x (S Muralikrishnan)
3. Towards OPAL & IPPL V 2.0 (A Adelmann)
4. pyOPAL (Ch Rogers)
5. Next releases (A Gsell)
6. Pressing open issues (all)