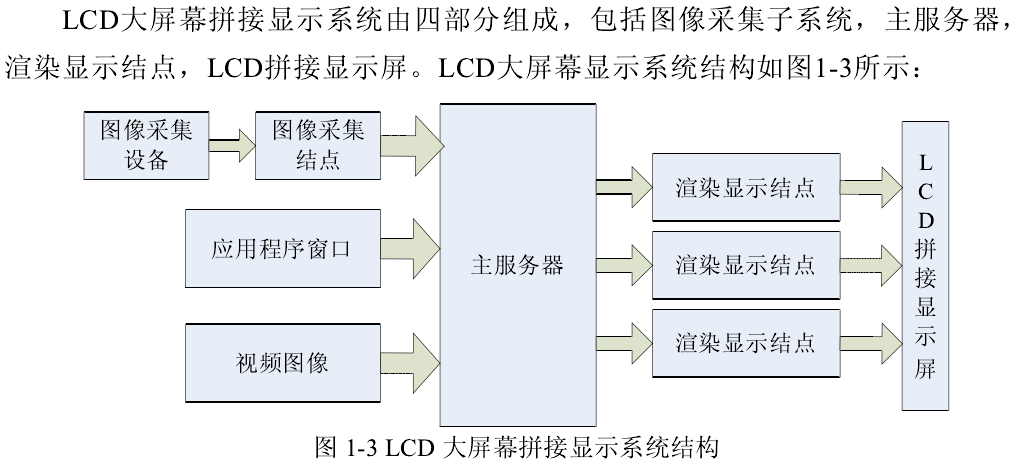
本周工作：

1. 看到和项目的系统结构相似的论文。

论文题目：**基于计算机集群的LCD大屏幕显示系统的设计与实现**



2.2.1 集群计算概述

计算机集群系统是指把一组计算机结点，按照某种网络结构用高性能通信网络连接起来，在并行操作系统的管理下，集群中的各计算机资源共享，协同工作，统一调度分配资源，以实现高效并行处理的系统。

2.2.2 并行渲染概述

在各种分布式的网络和局域网中，分布式图像显示系统对同步有着较高的要求。图像显示的实时性、准确性和系统的高效性都要求系统能进行实时数据通信。因此，数据的同步传输是分布式图像显示系统中的一个关键技术。

常用的同步方式有两种模式，一种是主从模式，一种是对等同步执行模式。 在主从模式中，所有输出内容的产生和处理都在主结点完成，然后由主结点发送到各从结点进行显示。每个从结点根据自己在拼接屏幕中的位置信息，显示接收到的图像内容。

对等同步执行模式是在各个显示结点中，均保存有一份完整图像的拷贝，每个结点负责读取、处理、渲染、显示本显示结点所需要显示的图像信息。在图像绘制之前，各显示结点的执行动作需要同步。

在两种同步模式中，主从模式结点之间传递的信息少，对网络性能要求不高，对内存需求较少，开发相对较容易。但是它的扩展性稍差，这种模式下，主结点承担了几乎所有的计算任务，从结点只是对接收到的图像数据进行显示，使得资源不能够充分得到利用，同时，随着系统的扩展，从结点数量的增加，主结点的处理能力和通信传输将成为整个系统的瓶颈。而相比主从模式而言，对等同步执行方式的扩展性很好，由于每个显示结点均保存有图像的拷贝，因此，各结点负责自己所需处理图像的处理工作，这样可以充分利用各结点的资源，但缺点是每个结点都需要同样内部和外部的存储空间，以便于对数据处理。

IEEE1588 定义了局域网络中可以使用的精确同步时钟协议(PTP 协议)，该协议适用于网络交流和本地计算。

IEEE1588 同步时钟协议规定，在进行时钟同步时，先由主设备通过多播形式发出时钟同步报文，所有与主设备在同一个域中的设备都将收到该同步报文。从设备收到同步报文后，根据同步报文中的时间戳和主时钟到从时钟的线路延时计算出与主时钟的偏差，对本地的时钟进行调整。

1. 思考有重叠区域情况的处理。



简单的处理方法，根据摄像机视场范围的外围矩形判断重叠的情况。

对于矩形，判断是否重叠的和重叠区域的计算比较简单。

在Qt当中提供图形对象的碰撞检测，主要提供一下三个函数：

bool QGraphicsItem::collidesWithItem(const [QGraphicsItem](qthelp://org.qt-project.qtwidgets.550/qtwidgets/qgraphicsitem.html#QGraphicsItem) \**other*, [Qt::ItemSelectionMode](qthelp://org.qt-project.qtwidgets.550/qtcore/qt.html#ItemSelectionMode-enum)*mode* = Qt::IntersectsItemShape) const

bool QGraphicsItem::collidesWithPath(const [QPainterPath](qthelp://org.qt-project.qtwidgets.550/qtgui/qpainterpath.html) &*path*, [Qt::ItemSelectionMode](qthelp://org.qt-project.qtwidgets.550/qtcore/qt.html#ItemSelectionMode-enum)*mode* = Qt::IntersectsItemShape) const

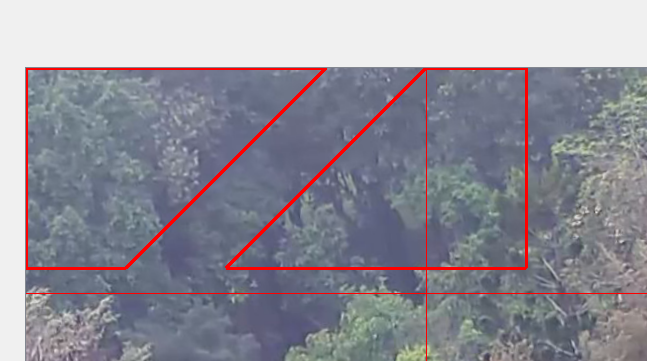
[QList](qthelp://org.qt-project.qtwidgets.550/qtcore/qlist.html)<[QGraphicsItem](qthelp://org.qt-project.qtwidgets.550/qtwidgets/qgraphicsitem.html#QGraphicsItem) \*> QGraphicsItem::collidingItems([Qt::ItemSelectionMode](qthelp://org.qt-project.qtwidgets.550/qtcore/qt.html#ItemSelectionMode-enum)*mode* = Qt::IntersectsItemShape) const

测试collidingItems()函数可用性。

编写计算两个矩形重叠区域矩形的函数：

输入两个矩形区域，运算得出矩形的重叠区域。

QRectF MyQMainWidget::intersectRect(QRectF rect1,QRectF rect2)



QPointF p1(400,0);

QPointF p2(500,0);

QPointF p3(500,200);

QPointF p4(200,200);

QPointF p5(0,0);

QPointF p6(300,0);

QPointF p7(100,200);

QPointF p8(0,200);

*collidesWithItem()*

Qt::IntersectsItemShape

Qt::IntersectsItemBoundingRect

false

true

通过函数计算重叠区域矩形：

QRectF(199.5,-0.5 100.854x201)



QPointF p1(350,100);

QPointF p2(500,0);

QPointF p3(500,200);

QPointF p4(200,200);

QPointF p5(0,0);

QPointF p6(300,0);

QPointF p7(350,99);

QPointF p8(0,200);

*collidesWithItem()*

Qt::IntersectsItemShape

Qt::IntersectsItemBoundingRect

true

true

通过函数计算重叠区域矩形：

QRectF(199.5,-0.416025 150.946x200.916)

接来下的工作：

1，重叠区域的切分，再转到重组节点的坐标系中。这个步骤可能受到重组节点的融合方法的影响。

2，优化整体步骤，如缩减多余的区域，不过会对代码改变较大。



